CONTENTS

CHAPTER 1

8085 PROCESSOR

1.1 INTRODUCTION TO MICROPROCESSORS

1.1.1 TERMS USED IN MICROPROCESSOR LITERATURE

Bit : A digit of the binary number or code is called a bit.

Nibble : The 4-bit (4-digit) binary number or code is called a nibble.

Byte : The 8-bit (8-digit) binary number or code is called a byte.

Word : The 16-bit (16-digit) binary number or code is called a word.

Double Word : The 32-bit (32-digit) binary number or code is called a double word.

Multiple Word : The 64, 128, ... bit/digit binary numbers or codes are called multiple words.

Data : The quantity (binary number/code) operated by an instruction of a program is called data. The size of data is specified as bit, byte, word,etc.

Address : Address is an identification number (in binary) for memory locations. The 8085 processor uses a 16-bit address for memory.

Memory Word Size (or Addressability) : The memory word size or addressability is the size of binary information that can be stored in a memory location. The memory word size for an 8085 processor-based system is 8-bit.

[Address and program codes in a microprocessor system are given in binary (i.e., as a combination of "0" and "1"). With n-bit binary we can generate 2^n *different binary codes or addresses.]*

Microprocessor : The microprocessor is a program-controlled semiconductor device (IC), which fetches instruction and data (from memory), decodes and executes instructions. It is used as CPU (**C**entral **P**rocessing **U**nit) in computers. The basic functional blocks of a microprocessor are ALU (**A**rithmetic **L**ogic **U**nit), an array of registers and a control unit. The microprocessor is identified with the size of data the ALU of the processor can work with at a time. The 8085 processor has a 8-bit ALU; hence, it is called a 8-bit processor. The 80486 processor has a 32-bit ALU; hence, it is called a 32-bit processor.

Bus : A bus is a group of conducting lines that carries data, address and control signals. Buses can be classified into Data bus, Address bus and Control bus.

The group of conducting lines that carries data is called a data bus.

The group of conducting lines that carries address is called an address bus.

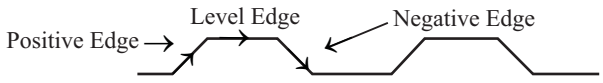
The group of conducting lines that carries control signals is called a control bus.

CPU Bus : The group of conducting lines that are directly connected to the microprocessor is called a CPU bus. In a CPU bus, the signals are multiplexed, i.e., more than one signal is passed through the same line but at different timings.

System Bus : The group of conducting lines that carries data, address and control signals in a microcomputer system is called System bus. Multiplexing is not allowed in a system bus.

[In microprocessor-based systems, each bit of information (data/address/control signal) is sent through a separate conducting line. Due to practical limitations, the manufacturers of microprocessors may provide multiplexed pins, i.e., one pin is used for more than one purpose. This leads to a multiplexed CPU bus. For example, in an 8086 processor, the address and data are sent through the same pins but at different timings. But when the system is formed, the multiplexed bus lines should be demultiplexed by using latches, ports, transceivers, etc. The demultiplexed bus lines are called system bus. In a system bus, separate conducting lines will be provided for each bit of data, address and control signals.]

Clock : A clock is a square wave used to synchronize various devices in the microprocessor and in the system. Every microprocessor system requires a clock for its functioning. The time taken for the microprocessor and the system to execute an instruction or program are measured only in terms of the time period of its clock.



A clock has three edges: rising edge (positive edge), level edge and falling edge (negative edge). The device is made sensitive to any one of the edges for better functioning (it means that the device will recognize the clock only when the edge is asserted or arrived).

Tristate Logic : Almost all the devices used in a microprocessor-based system use tristate logic. In devices with tristate logic, three logic levels will be available: **High** state, **Low** state and **High impedance** state.

The **high** and **low** level states are normal logic levels for data, address or control signals. The **high impedance** state is an electrical open-circuit condition. The **high impedance** state is provided to keep the device electrically isolated from the system. The tristate devices will normally remain in the **high impedance** state and their pins are physically connected in the system bus but electrically isolated. In the **high impedance** state, they cannot receive or send any signal or information. These devices are provided with chip enable/chip select pins. When the signal at this pin is asserted to the right level, they come out from the **high impedance** state to normal levels.

1.1.2 EVOLUTION OF MICROPROCESSORS

History tells us that it was the ancient Babylonians who first began using the abacus (a primitive calculator made of beads) in about 500 BC. This simple calculating machine eventually sparked the human mind into the development of calculating machines that use gears and wheels (Blaise Pascal in 1642). The giant computing machines of the 1940s and 1950s were constructed with relays and vacuum tubes. Next, the transistor and solid-state electronics were used to build the mighty computers of the 1960s. Finally, the advent of the **I**ntegrated **C**ircuit (IC) led to the development of the microprocessor and microprocessor-based computer systems.

In 1971, INTEL Corporation released the world's first microprocessor the INTEL 4004, a 4-bit microprocessor. It addresses 4096 memory locations of 4-bit word size. The instruction set consists of 45 different instructions. It is a monolithic IC employing large-scale integration in PMOS technology. The INTEL 4004 was soon followed by a variety of microprocessors, with most of the major semiconductor manufacturers producing one or more types.

First-Generation Microprocessors

The microprocessors introduced between 1971 and 1973 were the first-generation processors. They were designed using PMOS technology. This technology provided low cost, slow speed and low output currents and was not compatible with TTL (**T**ransistor **T**ransistor **L**ogic) levels.

The first-generation processors required a lot of additional support ICs to form a system, sometimes as high as 30 ICs. The 4-bit processors are provided with only 16 pins, but 8-bit and 16-bit processors are provided with 40 pins. Due to limitations of pins, the signals are multiplexed. A list of first-generation microprocessors are as follows:

1.	**INTEL 4004**	4-bit processors
2.	**INTEL 4040**	
3.	**FAIR CHILD PPS - 25**	
4.	**NATIONAL IMP - 4**	
5.	**ROCKWELL PPP - 4**	
6.	**MICRO SYSTEMS INTL. MC - 1**	
7.	**INTEL 8008**	8-bit processors
8.	**NATIONAL IMP - 8**	
9.	**ROCKWELL PPS - 8**	
10.	**AMI 7200**	
11.	**MOSTEK 5065**	
12.	**NATIONAL IMP/16**	16-bit processors
13.	**NATIONAL PACE**	

Second-Generation Microprocessors

The second-generation microprocessors appeared in 1973 and were manufactured using the NMOS technology. The NMOS technology offers faster speed and higher density than PMOS and it is TTL compatible. Some of the second-generation processors are as follows:

1. INTEL 8080
2. INTEL 8085
3. FAIRCHILD F - 8
4. MOTOROLA M6800
5. MOTOROLA M6809
6. NATIONAL CMP -8
7. RCA COSMAC
8. MOS TECH. 6500
9. SIGNETICS 2650
10. ZILOG Z80

} 8-bit processors

11. INTERSIL 6100
12. TOSHIBA TLCS - 12
13. TI TMS 9900

} 12-bit processors

14. DEC - W.D. MCP - 1600
15. GENERAL INSTRUMENT CP 1600
16. DATA GENERAL μN601

} 16-bit processors

Characteristics of Second-Generation Microprocessors

1. Larger chip size (170 × 200 mil). [1mil = 10^{-3} inch]
2. 40 pins.
3. More numbers of on-chip decoded timing signals.
4. The ability to address large memory spaces.
5. The ability to address more IO ports.
6. Faster operation.
7. More powerful instruction set.
8. A greater number of levels of subroutine nesting.
9. Better interrupt-handling capabilities.

Third-Generation Microprocessors

After 1978, the third-generation microprocessors were introduced. These are 16-bit processors and designed using HMOS (**H**igh density **MOS**) technology. Some of the third generation microprocessor are given below:

1. INTEL 8086
2. INTEL 8088
3. INTEL 80186
4. INTEL 80286
5. MOTOROLA 68000
6. MOTOROLA 68010
7. ZILOG Z8000
8. NATIONAL NS 16016
9. TEXAS INSTRUMENTS TMS 99000

The HMOS technology offers better **S**peed **P**ower **P**roduct (SPP) and higher packing density than NMOS.

Speed Power Product (SPP) = Speed × Power

Unit of SPP = Nanoseconds × Milliwatts

= Picojoules

1. Speed Power Product of HMOS is four times better than NMOS.
   SPP of NMOS = 4 picojoules (pJ)
   SPP of HMOS = 1 picojoules (pJ)
2. Circuit densities provided by HMOS are approximately twice those of NMOS.
   Packing density of NMOS = 1852.5 gates/mm^2
   Packing density of HMOS = 4128 gates/mm^2 (1 mm = 10^{-6} metre)

Characteristics of Third-Generation Microprocessors

- **Provided with 40/48/64 pins.**
- **High speed and very strong processing capability.**
- **Easier to program.**
- **Allow for dynamically relocatable programs.**
- **Size of internal registers are 8/16/32 bits.**
- **The processor has multiply/divide arithmetic hardware.**
- **Physical memory space is from 1 to 16 megabytes.**
- **The processor has segmented addresses and virtual memory features.**
- **More powerful interrupt-handling capabilities.**
- **Flexible IO port addressing.**
- **Different modes of operations (e.g., user and supervisor modes of M68000).**

Fourth-Generation Microprocessors

The fourth-generation microprocessors were introduced in the year 1980. These generation processors are 32-bit processors and are fabricated using the low-power version of the HMOS technology called HCMOS. These 32-bit microprocessors have increased sophistications that compete strongly with mainframes. Some of the fourth-generation microprocessors are given below:

1. **INTEL 80386**
2. **INTEL 80486**
3. **NATIONAL NS16032**
4. **MOTOROLA M68020**
5. **BELLMAC - 32**
6. **MOTOROLA M68030**
7. **MOTOROLA MC88100**

Characteristics of Fourth-Generation Microprocessors

1. **Physical memory space of 2^{24} bytes = 16 MB (megabytes).**
2. **Virtual memory space of 2^{40} bytes = 1TB (terabytes).**
3. **Floating-point hardware is incorporated.**
4. **Supports increased number of addressing modes.**

Fifth-Generation Microprocessors

In microprocessor technology, INTEL has taken a leading edge and is developing more and more new processors. The latest processor by INTEL is the **pentium** which is considered a fifth-generation processor. The pentium is a 32-bit processor with 64-bit data bus and is available in a wide range of clock speeds from 60 MHz to 3.2 GHz. With improvement in semiconductor technology, the processing speed of microprocessors has increased tremendously. The 8085 released in the year 1976 executes 0.5 **M**illion **I**nstructions **P**er **S**econd (0.5 MIPS). The 80486 executes 54 **M**illion **I**nstructions **P**er **S**econd. The pentium is optimized to execute two instructions in one clock period. Therefore, a pentium processor working at 1 GHz clock can execute 2000 **M**illion **I**nstructions **P**er **S**econd (2000 MIPS). The various processors released by INTEL are listed in Appendix I.

1.1.3 FUNCTIONAL BUILDING BLOCKS OF A MICROPROCESSOR

(AU, Nov/Dec' 19, 13 Marks)

A microprocessor is a programmable IC which is capable of performing arithmetic and logical operations. The basic functional block diagram of a microprocessor is shown in Fig. 1.1.

The basic functional blocks of a microprocessor are ALU, Flag register, Register array, **P**rogram **C**ounter (PC)/**I**nstruction **P**ointer (IP), Instruction decoding unit, and the Timing and Control unit.

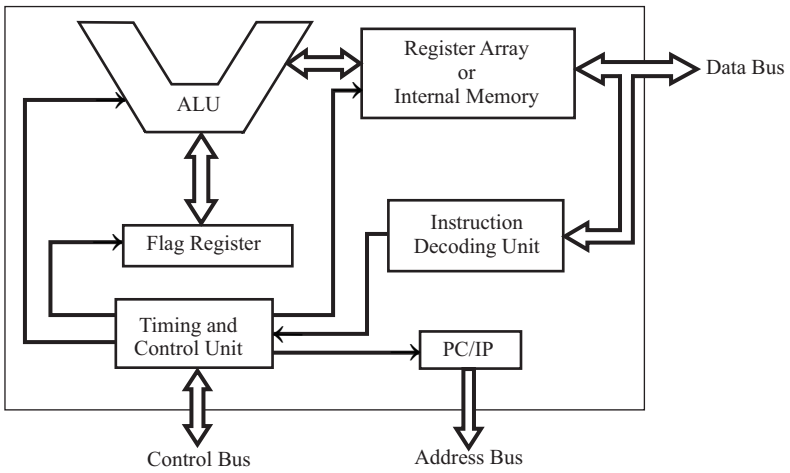


Fig. 1.1: Block diagram showing functional blocks of a microprocessor.

ALU is the computational unit of the microprocessor which performs arithmetic and logical operations on binary data. The various conditions of the result are stored as status bits called flags in the flag register. For example, consider sign flag. One of the bit positions of the flag register is called sign flag and it is used to store the status of sign of the result of the ALU operation (output data of ALU). If the result is negative then "1" is stored in the sign flag and if the result is positive then "0" is stored in the sign flag.

The register array is the internal storage device and so it is also called internal memory. The input data for ALU, the output data of ALU (result of computations) and any other binary information needed for processing are stored in the register array.

For any microprocessor, there will be a set of instructions given by its manufacturer. For doing any useful work with the microprocessor, we have to first write a program using these instructions, and store them in a memory device external to the microprocessor.

The instruction pointer generates the address of the instructions to be fetched from the memory and sends it through the address bus to the memory. The memory will send the instruction codes and data through the data bus. The instruction codes are decoded by the decoding unit and it sends information to the timing and control unit. The data is stored in the register array for processing by the ALU.

The control unit will generate the necessary control signals for internal and external operations of the microprocessor.

1.1.4 MICROPROCESSOR-BASED SYSTEM (ORGANIZATION OF A MICROCOMPUTER)

A microprocessor is a semiconductor device (or integrated circuit) manufactured by using the VLSI (**V**ery **L**arge **S**cale **I**ntegration) technique. It includes the ALU, the register arrays and the control circuit on a single chip. To perform a function or useful task we have to form a system by using the microprocessor as a CPU (**C**entral **P**rocessing **U**nit) and interfacing the memory, input and output devices to it. A system designed by using a microprocessor as its CPU is called a microcomputer or a single board microcomputer. A microprocessor-based system consists of a microprocessor as the CPU, semiconductor memories like EPROM and RAM, an input device, an output device and interfacing devices. The memories, input devices, output devices and interfacing devices are called peripherals.

The commonly used EPROM and static RAM in microcomputers are given below:

EPROM	Static RAM
INTEL 2708 (1 kB)	MOTOROLA 6208 (1kB)
INTEL 2716 (2 kB)	MOTOROLA 6216 (2kB)
INTEL 2732 (4kB)	MOTOROLA 6232 (4kB)
INTEL 2764 (8kB)	MOTOROLA 6264 (8kB)

Note: *kB refers to kilobytes.*

Popular input devices are keyboard, floppy disk, etc., and output devices are printer, LED and LCD displays, CRT monitor, etc.

The block diagram of a microprocessor-based system (or organization of microcomputer) is shown in Fig. 1.2. In this system the microprocessor is the master and all other peripherals are slaves. The master controls all the peripherals and initiates all operations.

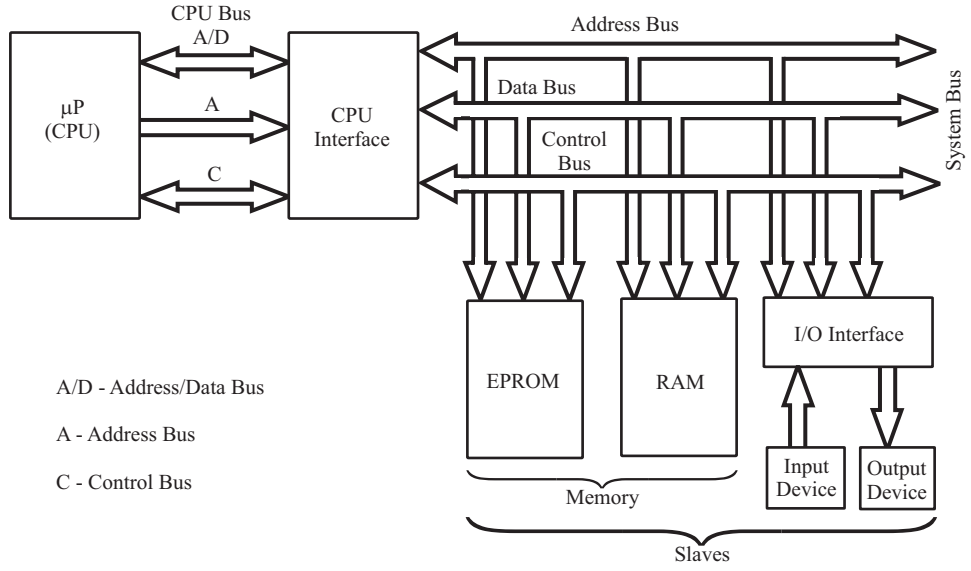


Fig. 1.2: Microprocessor-based system (organization of microcomputer).

Buses are groups of lines that carry data, addresses or control signals. The CPU bus has multiplexed lines, i.e., the same line is used to carry different signals. The CPU interface is provided to demultiplex the multiplexed lines to generate chip select signals and additional control signals. The system bus has separate lines for each signal.

All the slaves in the system are connected to the same system bus. But communication takes place between the master and one of the slaves at any one time. All the slaves have tristate logic and hence normally remain in **high impedance** state. The processor selects a slave by sending an address. When a slave is selected, it comes to the normal logic and communicates with the processor.

The EPROM memory is used to store permanent programs and data. The RAM memory is used to store temporary programs and data. The input device is used to enter the program, data and to operate the system. The output device is used for examining the results. Since the speed of IO devices does not match with the speed of the microprocessor, an interface device is provided between the system bus and the IO devices. Generally IO devices are slow devices.

The work done by the processor can be classified into the following three groups:

1. **Work done internal to the processor.**
2. **Work done external to the processor.**
3. **Operations initiated by the slaves or peripherals.**

The work done internal to the processor are additions, subtractions, logical operations, data transfer within registers, etc. The work done external to the processor are reading/writing the memory and reading/writing the IO devices or the peripherals. If the peripheral requires the attention of the master, then it can interrupt the master and initiate an operation.

The microprocessor is the master which controls all the activities of the system. To perform a specific job or task, the microprocessor has to execute a program stored in the memory. The program consists of a set of instructions stored in consecutive memory locations. In order to execute the program, the microprocessor issues address and control signals to fetch the instructions and data from the memory one by one. After fetching each instruction it decodes the instructions and performs the task specified by the instruction.

1.1.5 CONCEPT OF MULTIPLEXING IN MICROPROCESSOR

Multiplexing is transferring different information at different well-defined times through the same lines. A group of such lines is called a multiplexed bus. The result of multiplexing is that fewer pins are required for microprocessors to communicate with the outside world.

Due to the pin number limitations, most microprocessors cannot provide simultaneously similar lines (such as address, data, status signals, etc.). Hence multiplexing of one or more of these buses is performed. Most often data lines are multiplexed with some or all address lines to form an address/data bus. (e.g., In 8085 the lower 8-address lines are multiplexed with data lines.) The status signals emitted by the microprocessor are sometimes multiplexed either with the data lines (as done in the INTEL 8080A) or with some of the address lines (as done in the INTEL 8086).

Whenever multiplexing is used, the CPU interface of the system must include the necessary hardware to demultiplex those lines to produce separate address, data and control buses required for the system. Demultiplexing of a multiplexed bus can be handled either at the CPU interface or locally at appropriate points in the system. Besides a slower system operation, a multiplexed bus also results in additional interface hardware requirements.

Demultiplexing of Address/Data Lines in 8085 Processor

In order to demultiplex the address/data lines (of the processor), the processor provides a signal called the ALE (**A**ddress **L**atch **E**nable). The ALE is asserted **high** and then **low** by the processor at the beginning of every machine cycle. At the same time the low byte address is given out through the AD_0 - AD_7 lines. The demultiplexing of address/data lines using an 8-bit D-latch 74LS373 is shown in Fig. 1.3.

The ALE is connected to the enable pin (EN) of an external 8-bit latch. When the ALE is asserted **high** and then **low**, the addresses are latched into the output lines of the latch. It holds the low byte of the address until the next machine cycle. After latching the address, the AD_0 - AD_7 lines are free for data transfer. The first T-state of every machine cycle is used for address latching in 8085 and the remaining T-states are used for reading or writing operation.

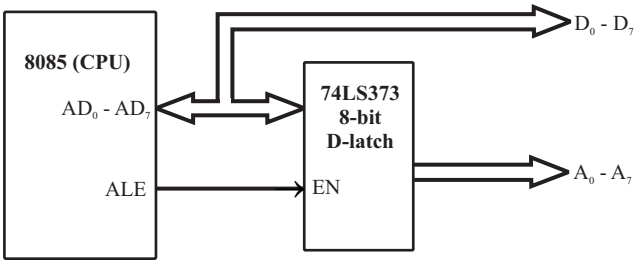


Fig. 1.3: Demultiplexing of address and data lines in an 8085 processor.

1.2 HARDWARE ARCHITECTURE AND PINOUTS

The INTEL 8085 is an 8-bit microprocessor released in the year 1976. The 8085 was originally designed using NMOS technology but now it is manufactured using HMOS technology and contains approximately 6500 transistors. The 8085 is packed in a 40-pin DIP (**D**ual **I**n-line **P**ackage) and requires a single 5V supply.

The 8085 has an internal clock oscillator. It generates a clock signal internally and divides by two for use as internal clock. This internal clock is also given out through the CLK pin for the clock requirement of peripheral devices.

The NMOS 8085 is available in two versions: 8085A and 8085A-2, with a maximum internal clock frequency of 3.03 MHz and 5 MHz respectively. The enhanced version of the 8085 is designed with HMOS transistors. It is available in three versions: 8085AH, 8085AH-2 and 8085AH-1 with maximum internal clock of 3 MHz, 5 MHz and 6 MHz respectively.

The basic data size of an 8085 is 8-bit. Therefore the memory word size of the memories interfaced with a 8085 processor is also 8-bit or byte. The 8085 uses a 16-bit address to access memory and hence it can address up to $2^{16} = 65{,}536_{10} = 64$ k memory locations. Since, one byte of information can be stored in one memory location, the maximum memory capacity of an 8085-based system is 64 kilobytes. For accessing IO-mapped devices, the 8085 uses a separate 8-bit address and so it can generate $2^8 = 256_{10}$ IO addresses.

1.2.1 PIN CONFIGURATION OF 8085

The pin configuration of an 8085 microprocessor is shown in Fig. 1.4. The signals of the 8085 are listed in Table 1.1. The 8085 has 8 pins AD_0 to AD_7 for data transfer, which are multiplexed with low byte of address. The 8085 provides a signal ALE (**A**ddress **L**atch **E**nable) to demultiplex the low byte address and data using an external latch. The demultiplexing of address and data lines in an 8085 is shown in Fig. 1.3 in Section 1.1.5.

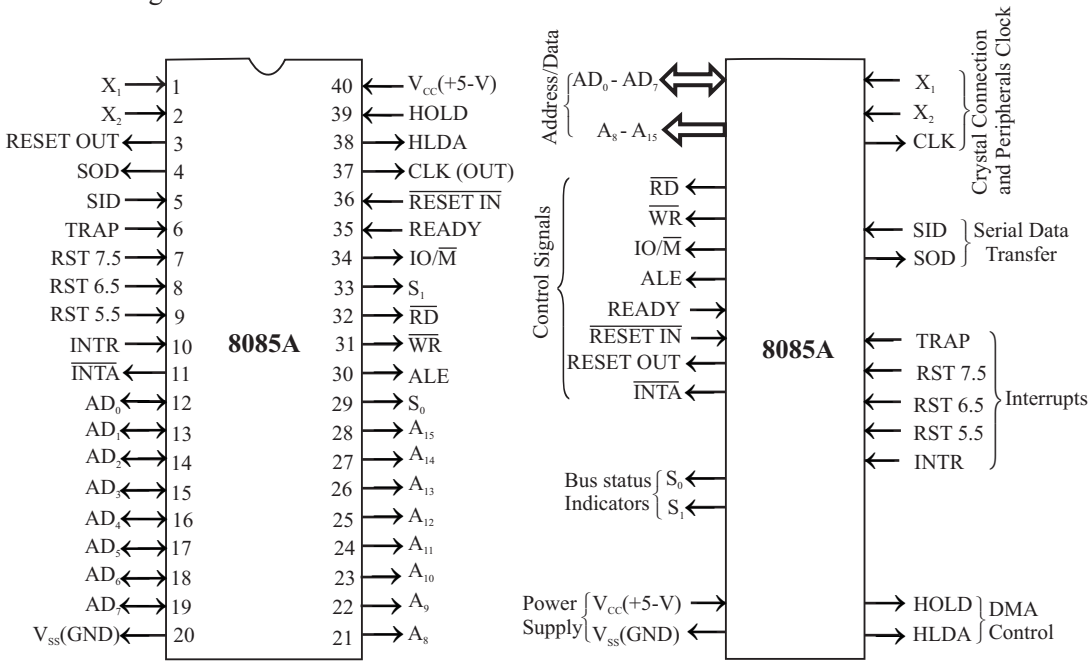


Fig. 1.4: 8085 microprocessor signals and pin assignment.

During memory access, the 16-bit memory address are output on AD_0 to AD_7 and A_8 to A_{15} lines. During IO access of IO-mapped devices the 8-bit IO address are output on both AD_0 to AD_7 and A_8 to A_{15} lines. The 8085 processor differentiates the memory and IO address using the signal IO/$\overline{M}$. When the processor outputs a memory address, the IO/$\overline{M}$ is asserted **low** and when the processor outputs an IO address, the IO/$\overline{M}$ is asserted **high**.

The $\overline{RD}$ signal is asserted **low** by the processor during a memory or IO read operation. The $\overline{WR}$ signal is asserted **low** by the processor during a memory or IO write operation. The S_0 and S_1 are bus status indicators. The output signals on these lines during various bus activity (or machine cycles) are listed in Table 1.2.

Table 1.1: 8085 Signal Description Summary

Pin Name	Description	Type
AD_0 - AD_7	Address/Data	Bidirectional, Tristate
A_8 - A_{15}	Address	Output, Tristate
ALE	Address latch enable	Output, Tristate
$\overline{RD}$	Read control	Output, Tristate
$\overline{WR}$	Write control	Output, Tristate
IO/$\overline{M}$	IO or memory indicator	Output, Tristate
S_0, S_1	Bus state indicators	Output
READY	Wait state request	Input
SID	Serial input data	Input
SOD	Serial output data	Output
HOLD	Hold request	Input
HLDA	Hold acknowledge	Output
INTR	Interrupt request	Input
TRAP	Nonmaskable interrupt request	Input
RST 5.5	Hardware vectored interrupt request	Input
RST 6.5	Hardware vectored interrupt request	Input
RST 7.5	Hardware vectored interrupt request	Input
$\overline{INTA}$	Interrupt acknowledge	Output
$\overline{RESET\ IN}$	System reset	Input
$\overline{RESET\ OUT}$	Peripherals reset	Output
X_1, X_2	Crystal or RC connection	Input
CLK (OUT)	Clock signal	Output
V_{cc}	+5 V	Power supply
V_{ss}	Ground	Power supply

Note: A overbar on the signal, indicates that it is active low. (i.e., the signal is normally high and when the signal is activated it is low).

Table 1.2: Bus Status Signals

IO/$\overline{M}$	S_1	S_0	Operation performed by the 8085
0	0	1	Memory write
0	1	0	Memory read
1	0	1	IO write
1	1	0	IO read
0	1	1	Opcode fetch
1	1	1	Interrupt acknowledge

READY is an input signal that can be used by slow peripherals to get extra time in order to communicate with the 8085. The 8085 will work only when READY is tied to logic **high**. Whenever READY is tied to logic **low**, the 8085 will enter a wait state. When the system has slow peripheral devices, additional hardware is provided in the system to make the READY input **low** during the required extra time while executing a machine cycle, so that the processor will remain in wait state during this extra time.

The HOLD and HLDA signals are used for **D**irect **M**emory **A**ccess (DMA) type of data transfer. This type of data transfers are achieved by employing a DMA controller in the system. When DMA is required, the DMA controller will place a **high** signal on the HOLD pin of the 8085. When the HOLD input is asserted **high**, the processor will enter a wait state and drive all its tristate pins to a **high impedance** state and send an acknowledgement signal to the DMA controller through the HLDA pin. Upon receiving the acknowledgement signal, the DMA controller will take control of the bus and perform DMA transfer and at the end it asserts HOLD signal **low**. When HOLD is asserted **low** the processor will resume its execution.

The 8085 has five interrupt pins. The order of priority of the interrupts is TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR. The interrupts TRAP, RST 7.5, RST 6.5 and RST 5.5 are hardware vectored interrupts and are enabled by appropriate signals at the appropriate pins of the 8085. When a vectored interrupt is enabled and if it is accepted then the program execution branches to the vector addresses specified by INTEL. The interrupts RST 7.5, RST 6.5 and RST 5.5 are maskable interrupts by software.

The INTR is enabled by appropriate signals at its pin. In order to service the INTR, one of the eight opcodes (RST 0 to RST 7) has to be provided on the AD_0 - AD_7 bus by external logic. The 8085 then executes this instruction and vectors to the appropriate address to service the interrupt.The vector address for an interrupt RST n is given by $\mathbf{(08 \times n)_H}$. The vector addresses of the interrrupts of 8085 are listed in Table 1.3. (The interrupt TRAP is RST 4.5.)

Table 1.3: Vector Addresses of Interrupts

Interrupt	Vector address
RST 0	0000_H
RST 1	0008_H
RST 2	0010_H
RST 3	0018_H
RST 4	0020_H
TRAP	0024_H

Interrupt	Vector address
RST 5	0028_H
RST 5.5	$002C_H$
RST 6	0030_H
RST 6.5	0034_H
RST 7	0038_H
RST 7.5	$003C_H$

The 8085 has the clock generation circuit on the chip but an external quartz crystal or LC circuit or RC circuit should be connected at the pins X_1 and X_2. The frequency at X_1 and X_2 is divided by two internally and used as an internal clock. The frequency of the output clock signal at the CLK(OUT) pin is same as that of the internal clock.

$\overline{\text{RESET IN}}$ is the system reset input signal and it is used to bring the processor to a known state. For proper reset, the $\overline{\text{RESET IN}}$ pin should be held **low** for at least three clock periods. When pin is asserted **low**, the program counter, instruction register, interrupt mask bits and all internal registers are cleared/reset. Also the RESET OUT signal is asserted **high** to clear/reset all the peripheral devices in the system. After a reset, the content of the program counter will be 0000_H and so the processor will start executing the program stored at 0000_H.

The pins SID and SOD can be used for serial data communication between the 8085 and any serial device under software control.

Driving X_1 and X_2 Inputs

The X_1 and X_2 pins of an 8085 processor are provided to connect an external quartz crystal or LC circuit. It can also be driven by an RC circuit or an external clock source. This connection is necessary for the internal oscillator to generate the clock signal for the processor. An oscillator consists of an amplifier and a feedback circuit. The feedback circuit of an oscillator can be of RC type, LC type or quartz crystal (a quartz crystal is electrically equivalent to an RLC circuit.) Also the feedback circuit decides the frequency of the signal generated by the oscillator.

In an 8085 processor, the oscillator circuit is provided internally except the feedback circuit. This feature facilitates the system designer to choose his own frequency for clock signals. But this frequency should not exceed the maximum clock frequency specified by the manufacturer. Another reason for keeping feedback circuit external to the processor is that the high Q circuits (quartz crystal or large values of L) cannot be fabricated by IC technology.

In an 8085, the frequency generated by the oscillator circuit will be double that of the internal clock frequency. (The maximum clock frequencies specified by the manufacturer are internal clock frequencies.) In other words, the frequency at X_1 - X_2 pins of an 8085 is divided by two internally. This means that in order to obtain an internal clock of 3.03 MHz, a clock source of 6.06 MHz must be connected to X_1 - X_2. (Crystal/LC/RC should be designed for double the internal frequency.)

Quartz crystals are the best choice for connecting at X_1 - X_2, because they are less expensive, highly stable, have a large Q, occupy a very small space and frequencies do not drift with ageing. For crystals with less than 4 MHz, a capacitor of 20 pF should be connected between X_2 and ground to ensure the starting up of the crystal at the right frequency.

When an LC circuit is used, the value of L_{ext} and C_{ext} can be chosen using the formula,

$$f = \frac{1}{2\pi L_{ext}(C_{ext} + C_{int})}$$

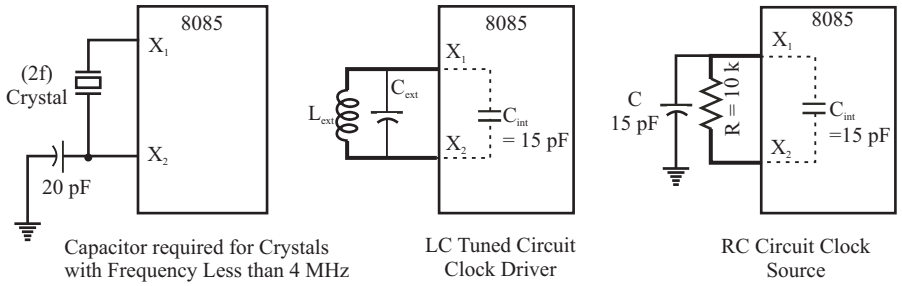


Fig. 1.5: Clock driver circuits for an 8085.

To minimize the variations in frequency, it is recommended that the value for C_{ext} should be chosen which is twice that of C_{int} or 30 pF. The use of LC circuit is not recommended for external frequencies higher than 5 MHz.

An RC circuit may also be used as the clock source for the 8085A if an accurate clock frequency is of no concern. Its advantage is the low component cost. The values shown in Fig. 1.5 are for generating an approximate external frequency of 3 MHz. Note that frequencies higher or lower than 3 MHz should not be attempted on this circuit.

1.2.2 HARDWARE ARCHITECTURE OF 8085

The architecture of an 8085 is shown in Fig. 1.7. The 8085 includes an ALU, a timing and control unit, a instruction register and a decoder, a register array, an interrupt control and a serial IO control.

The ALU performs the arithmetic and logical operations. The operations performed by the ALU of an 8085 are **addition, subtraction, increment, decrement, logical AND, OR, EXCLUSIVE-OR, compare, complement and left/right shift.** The accumulator and temporary register are used to hold the data during an arithmetic/logical operation. After an operation the result is stored in the accumulator and the flags are set or reset according to the result of the operation. The accumulator and flag register together is called the **P**rogram **S**tatus **W**ord (PSW).

There are five flags in an 8085: **Sign Flag (SF), Zero Flag (ZF), Auxiliary Carry Flag (AF), Parity Flag (PF) and Carry Flag (CF).** The bit positions reserved for these flags in the flag register are shown in Fig. 1.6.

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SF	ZF		AF		PF		CF

Fig. 1.6: Bit positions of various flags in the flag register of 8085.

After an ALU operation if the most significant bit of the result is 1, the sign flag is set. The zero flag is set if the ALU operation results in zero and it is reset if the result is nonzero. In an arithmetic operation, when a carry is generated by the lower nibble, the auxiliary carry flag is set. After an arithmetic or logical operation if the result has an even number of 1's, the parity flag is set, otherwise it is reset.

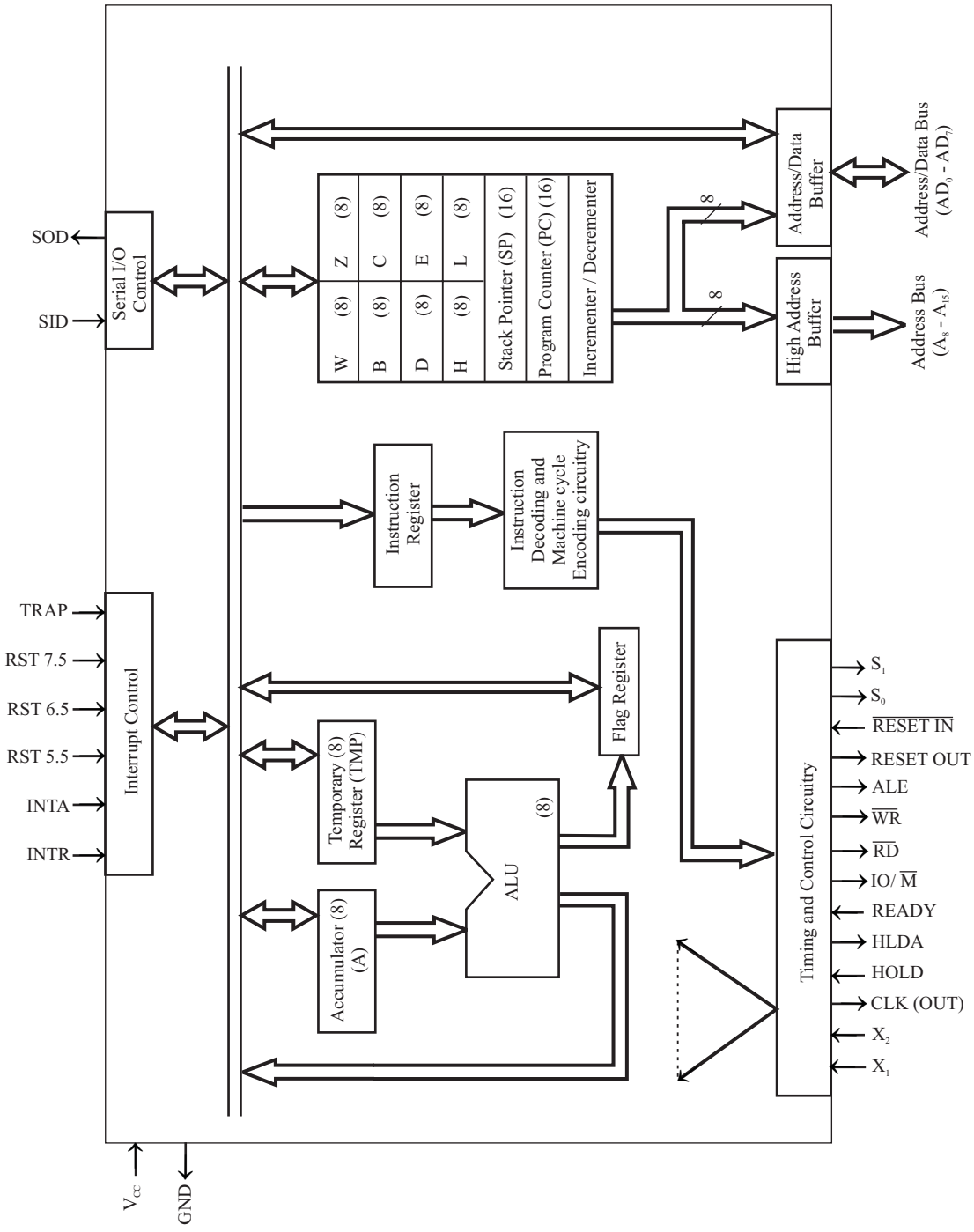


Fig. 1.7: Architecture of Intel 8085 microprocessor.

If an arithmetic operation results in a carry, the carry flag is set, otherwise it is reset. Among the five flags, the AF flag is used internally for BCD arithmetic and other four flags can be used by the programmer to check the conditions of the result of an operation.

The **timing and control uni**t synchronizes all the microprocessor operations with the clock, and generates the control signals necessary for communication between the microprocessor and the peripherals.

When an instruction is fetched from the memory it is placed in the instruction register. Then it is decoded and encoded into various machine cycles. A part from the **Accumulator** (A-register) there are six general purpose programmable registers B, C, D, E, H and L. They can be used as 8-bit registers or paired to store 16-bit data. The allowed pairs are BC, DE and HL. The temporary registers TMP, W and Z cannot be used by the programmer.

The **S**tack **P**ointer SP holds the address of the stack top. The stack is a sequence of RAM memory locations defined by the programmer. The stack is used to save the content of the registers during the execution of a program.

The **P**rogram **C**ounter (PC) keeps track of program execution. To execute a program the starting address of the program is loaded in the program counter. The PC sends out an address to fetch a byte of instruction from memory and increment its content automatically. Hence when a byte of instruction is fetched, the PC holds the address of the next byte of the instruction or the next instruction.

Instruction Execution and Data Flow in 8085

The program instructions are stored in the memory, which is an external device. In order to execute a program in an 8085, the starting address of the program should be loaded in the program counter. The 8085 outputs the contents of the program counter to the address bus and asserts the read control signal **low**. Also, the program counter is incremented.

The address and the read control signal enables the memory to output the content of the memory location on the data bus. Now the content of the data bus is the opcode of an instruction. The read control signal is made **high** by the timing and control unit after a specified time. At the rising edge of read control signals, the opcode is latched into the microprocessor internal bus and placed in the instruction register.

The instruction decoding unit decodes the instructions and provides information to the timing and control unit to take further action.

1.3 MEMORY ORGANIZATION

A memory unit is an integral part of any microcomputer system and its primary purpose is to store programs and data. In a broad sense, a microcomputer memory system can be logically divided into three groups. They are as follows:

- **Processor memory**
- **Primary or main memory**
- **Secondary memory**

Processor memory refers to registers inside the microprocessor. These registers are used to hold data and results temporarily when computation is in progress. Since the registers of the processor are fabricated using the same technology as that of a microprocessor, there is no speed disparity between these registers and a microprocessor. However, the cost involved in this approach forces a manufacturer to include only a few registers in the microprocessor.

Primary or main memory refers to the storage area which can be directly accessed by the microprocessor. Therefore, all programs and data must be stored only in primary memory prior to execution. In primary memory the access time should be compatible with the read/write time of the processor. Therefore, only semiconductor memories are used as primary memories and they (the latest versions) are fabricated using CMOS technology. Primary memory normally includes ROM, EPROM, static RAM, DRAM and NVRAM.

Secondary memory refers to the storage medium which comprises of slow devices such as magnetic tapes and disks (hard disk, floppy disc and **C**ompact **D**isc (CD)). They are called as auxiliary or backup storage devices. These devices are used to hold large data files and huge programs such as operating systems, compilers, data bases, permanent programs, etc. The microcomputer system copies the required programs and data from secondary memory to main memory and work directly with main memory only.

1.3.1 INTERFACING STATIC RAM AND EPROM

The primary function of memory interfacing is that the microprocessor should be able to read from and write into a set of semiconductor memory IC chips. Generally, EPROM is interfaced for read operations and RAM is interfaced for read and write operations. The procedure for interfacing SRAM for read/write operation and EPROM for read operation are similar. So, they are dealt commonly in this section.

In order to perform the read/write operation the memory access time should be less than the read/write time of processor, chip select signals should be generated for selecting a particular memory IC, suitable control signals have to be generated for read/write operation and a specific address should be allotted to each memory location.

Hence, memory interfacing deals with choosing memories with suitable access time, designing address decoding circuit to generate chip select signals, generating control signals for read/write operation and allocation of addresses to various memory ICs and their locations.

Typical EPROM and Static RAM

A typical semiconductor memory IC will have **n** address pins, **m** data pins (or output pins) and a minimum of two power supply pins (one for connecting required supply voltage (V_{CC}) and the other for connecting ground). The control signals needed for static RAM are chip select (chip enable), read control (output enable) and write control (write enable). The control signals needed for read operation in EPROM are chip select (chip enable) and read control (output enable). A typical static RAM and EPROM are shown in Fig. 1.8 and Fig. 1.9 respectively.

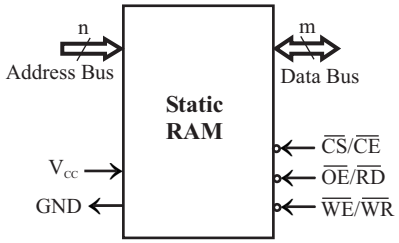


Fig. 1.8: A typical static RAM IC.

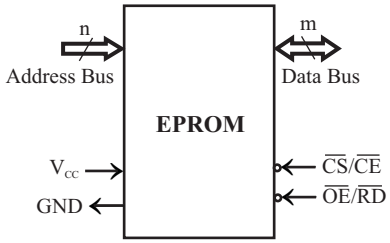


Fig. 1.9: A typical EPROM IC in read mode.

$\overline{CS}/\overline{CE}$ - Chip Select (or Chip Enable) ; $\overline{OE}/\overline{RD}$ - Output Enable (or Read Control)
$\overline{WE}/\overline{WR}$ - Write Enable (or Write Control)

Note:	*The pins of EPROM are redefined for write operation. An EPROM requires a different hardware setup and high supply voltage for write operation.*

Memory Capacity

A semiconductor memory IC will have **n** address pins and **m** data pins. Such a memory has $\mathbf{2^n}$ locations and each location can store **m**-bit data. The size of data stored in each memory location is called memory word size. In INTEL 8085-based systems normally memories with word size of 1-byte are used. (But we can even interface memories with word size 1-bit, 2-bit and 4-bit.)

The memory capacity is specified in kilobytes. If the memory IC has **m** data pins and **n** address pins, then the memory IC will have a capacity of $2^n \times m$ bits. When m = 8, the memory capacity is 2^n bytes. One kilobyte is $1024_{10} (= 400_H)$ bytes. The relation between address pins and capacity of memory ICs are listed in Table 1.4.

Table 1.4: Relation between Number of Address Pins and Memory Capacity

Number of address pins	Memory capacity			Range of address in hexa
	in decimal	in kilo	in hexa	
10	$2^{10} = 1024$	1k	400	000 to 3FF
11	$2^{11} = 2 \times 2^{10} = 2048$	2k	800	000 to 7FF
12	$2^{12} = 2^2 \times 2^{10} = 4 \times 2^{10} = 4096$	4k	1000	000 to FFF
13	$2^{13} = 2^3 \times 2^{10} = 8 \times 2^{10} = 8192$	8k	2000	0000 to 1FFF
14	$2^{14} = 2^4 \times 2^{10} = 16 \times 2^{10} = 16384$	16k	4000	0000 to 3FFF
15	$2^{15} = 2^5 \times 2^{10} = 32 \times 2^{10} = 32768$	32k	8000	0000 to 7FFF
16	$2^{16} = 2^6 \times 2^{10} = 64 \times 2^{10} = 65536$	64k	10000	0000 to FFFF

Choice of Memory ICs and Address Allocation

The memory requirement of a system depends on the application for which it is designed. A system designer has a variety of choices for choosing memory ICs. The total memory requirement can be realized in a single IC or in multiple ICs.

The total memory requirement of the system will be split between EPROM and RAM memories. The EPROM memories are used for storing monitor programs, other permanent programs and data. The RAM memories are used for stack operations, temporary program and data storage.

Popular EPROM and static RAM ICs with 8085 systems and their capacity are listed here. Table 1.5 shows the number of address pins and data pins available on these ICs.

EPROM

2708 (1k × 8 = 8 kilobits/1kB)

2716 (2 k × 8 = 16 kilobits/2 kB)

2732 (4 k × 8 = 32 kilobits/4 kB)

2764 (8 k × 8 = 64 kilobits/8 kB)

27256 (32 k × 8 = 256 kilobits/32 kB)

27512 (64 k × 8 = 512 kilobits/64 kB)

Static RAM

6208 (1k × 8 = 8 kilobits/1kB)

6216 (2 k × 8 = 16 kilobits/2 kB)

6232 (4 k × 8 = 32 kilobits/4 kB)

6264 (8 k × 8 = 64 kilobits/8 kB)

62256 (32 k × 8 = 256 kilobits/32 kB)

62512 (64 k × 8 = 512 kilobits/64 kB)

Note: *In this book kB refers to kilobytes.*

Table 1.5: Number of Address and Data Pins in Memory ICs

Memory IC EPROM/RAM	Capacity	Number of address pins	Number of data pins
2708/6208	1 kB	10	8
2716/6216	2 kB	11	8
2732/6232	4 kB	12	8
2764/6264	8 kB	13	8
27256/62256	32 kB	15	8
27512/62512	64 kB	16	8

Note: *16 kB memory is not available as a standard product.*

Generation of Chip Select Signals

Decoders are used to generate chip select signals. The 2-to-4 decoder will give four chip select signals. The 3-to-8 decoder will give eight chip select signals. The 4-to-16 decoder will give sixteen chip select signals.

Decoder is a logic circuit that identifies each combination of the signals present at its input. Decoders have **n** input lines and $\mathbf{2^n}$ output lines. In logic **low** decoder, at any one time one of the 2^n outputs will remain **low** and all other outputs will remain **high**.

The output which remains **low** depends on the input signal. Hence if the decoder outputs are connected to chip select pins of ICs in the microprocessor system at any one time, only one chip will be selected. The input to the decoders are unused address lines or high order address lines.

While interfacing memories, low order address lines are connected to memory ICs. The remaining unused address lines (or high order address lines) are connected to the input of the decoder. The outputs of the decoder are connected to CS or CE pins of memory ICs.

In a microprocessor-based system, all the memory ICs and peripheral ICs are connected to a common system bus. Therefore, the data, address and control lines are connected to all the slaves (memory/peripheral ICs). But all the slaves remain in **high impedance** state. So, they cannot communicate with the master (processor) through bus (i.e., they are physically connected but electrically isolated).

When the address is given out by the processor for read/write operation, only one of the memory ICs is selected and the selected memory IC will come to normal logic. The selection logic depends on address decoding logic. All other memory ICs will remain in **high impedance** state. So, they are electrically isolated from the system. The read/write operation is performed by the processor with the selected memory IC.

Decoder

Popular decoders used in the microprocessor-based system are 74LS138 and 74LS139. The 74LS138 is a 3-to-8 decoder and 74LS139 is dual 2-to-4 decoder.

The 74LS138 decoder consists of 3-input lines, 8-output lines (logic **low**) and three enables or ground. In the three enables, two are logic **low** and one is a logic **high** enable. The pin configuration of 3-to-8 decoder (74LS138) is shown in Fig. 1.10. The truth table of the decoder is given in Table 1.6.

The 74LS139 decoder consists of two numbers of 2-to-4 decoder packed in a single IC package. Each decoder has two input pins, four output lines and a logic **low** enable. The pin configuration of 74LS139 is shown in Fig. 1.11. The truth table of 2-to-4 decoder is given in Table 1.7. In the 74LS139 each decoder can work independently.

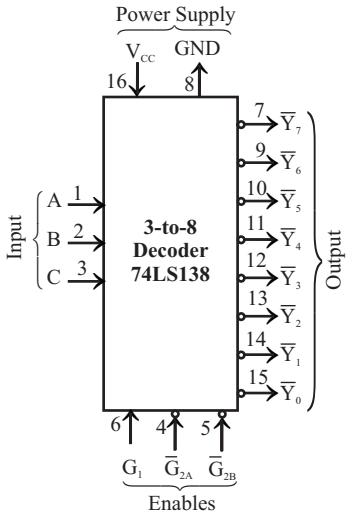


Fig. 1.10: Signals of 74LS138.

Table 1.6: Truth Table of 3-to-8 Decoder

Enables			Input			Output							
G_1	$\overline{G}_{2A}$	$\overline{G}_{2B}$	C	B	A	$\overline{Y}_7$	$\overline{Y}_6$	$\overline{Y}_5$	$\overline{Y}_4$	$\overline{Y}_3$	$\overline{Y}_2$	$\overline{Y}_1$	$\overline{Y}_0$
1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	1	0	1	1	1	1	1
1	0	0	1	1	0	1	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1	1	1
0	1	1	X	X	X	H	H	H	H	H	H	H	H

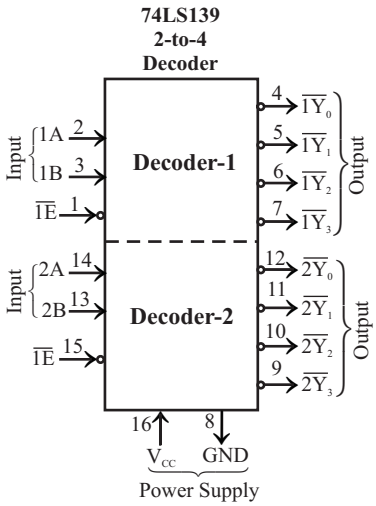


Fig. 1.11: Signals of 74LS139.

Table 1.7: Truth Table of The 2-to-4 Decoder

Enable	Input		Output			
$\overline{E}$	B	A	$\overline{Y}_3$	$\overline{Y}_2$	$\overline{Y}_1$	$\overline{Y}_0$
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1
1	X	X	H	H	H	H

1.3.2 MEMORY ORGANIZATION IN 8085-BASED SYSTEM

A microprocessor-based system requires both EPROM and RAM. Hence the available memory space has to be divided between EPROM and RAM. The 8085 has 64 kB of addressable memory space and allotting this address space for EPROM and RAM depends on the system designer as well as the application for which the system is designed.

Some systems may require large memory space. So, the full memory space is utilized. But in some systems the memory requirement may be less and in this case the full memory space will not be utilized. When the full memory space is not utilized, the unused memory addresses can be used for addressing IO devices. Such IO devices are called memory-mapped IO devices and they can be accessed similar to that of a memory device.

In 8085 system, the EPROM is mapped at the beginning of memory space (i.e., 0000_H address is allotted to EPROM memory location). Whenever the power supply is switched ON, the microprocessor chip will be reset. This power-on reset will be implemented by the system designer. When the processor is reset all the internal registers, flag register and program counter will be cleared. Hence, after a reset, the program counter will have an address 0000_H and so the processor starts fetching and executing the instruction stored at 0000_H.

The system designer will store the monitor program starting from the address 0000_H. The monitor program should be executed to initialize system peripherals whenever the system is switched ON. To enable automatic execution of monitor program, whenever the system is switched ON, the EPROM should be mapped from 0000_H location in 8085-based system. Monitor program is a permanent program written by the system designer to take care of system initializations. System initializations includes the following :

(a) Programming 8279 for keyboard scanning and display refreshing.

(b) Programming peripheral ICs 8259, 8257, 8255, 8251, 8254, etc.

(c) Initializing stack.

(d) Display a message on display (output) device.

(e) Initializing interrupt vector table.

Note: 8279 - Programmable keyboard/display controller. 8257 - DMA controller.
8259 - Programmable interrupt controller. 8251 USAR
8255 - Programmable peripheral interface. 8254 - Programmable timer.

The total address space and its allocation is shown in Fig. 1.12.

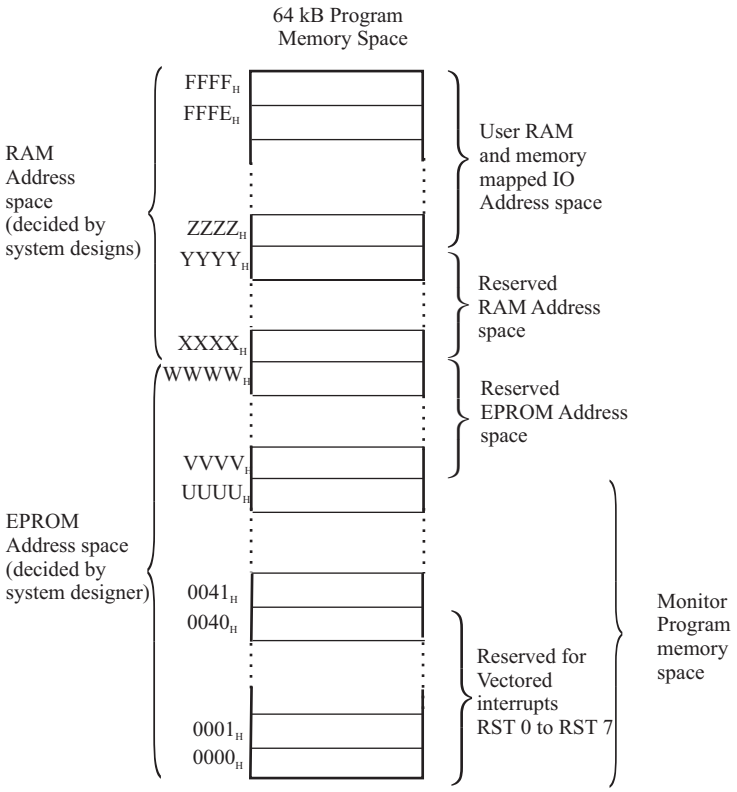


Fig. 1.12: Memory Address Space in 8085 Microprocessor.

The required EPROM memory capacity of the system can be implemented in one IC or in multiple ICs. Similarly the RAM capacity of the system can be implemented in one IC or in multiple ICs. This choice depends on the availability of memory IC and the system designer. Some examples of memory organizations for 8085 microprocessor-based system are discussed in this section.

Consider a system in which the full memory space 64 kB is utilized for EPROM memory. In this system the entire 16 address lines of the processor are connected to address input pins of memory IC in order to address the internal locations of memory and **C**hip **S**elect (CS) pin of EPROM is permanently tied to logic **low** (i.e., tied to ground) as shown in Fig. 1.13. Now the range of address for EPROM is 0000_H to $FFFF_H$.

Consider a system in which the available 64 kB memory space is equally divided between EPROM and RAM. Let us implement 32 kB memory capacity of EPROM using single IC 27256. Similarly, 32 kB RAM capacity is implemented using single IC 62256. The 32 kB memory requires 15 address lines and so the address lines A_0 - A_{14} of the processor are connected to 15 address pins of both EPROM and RAM as shown in Fig. 1.13.

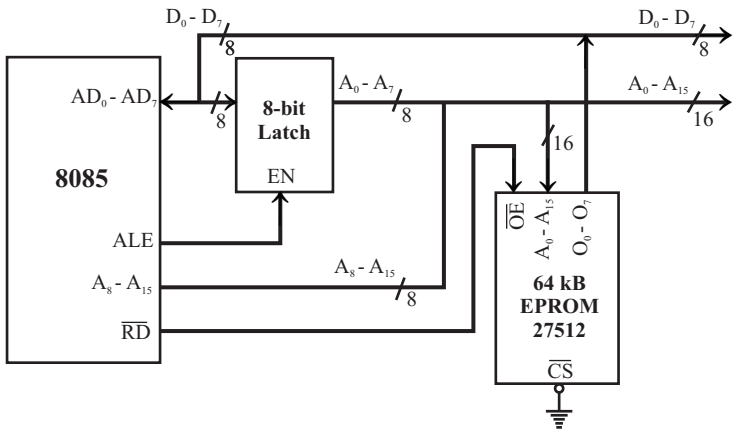


Fig. 1.13: Example of implementing 64 kB EPROM in the 8085 system.

The unused address line A_{15} is used as a chip select signal for selecting either EPROM or RAM. The A_{15} line is directly connected to the CS pin of EPROM and it is inverted and connected to CS pin of RAM. Therefore, the EPROM is selected when $A_{15} = 0$ and RAM is selected when $A_{15} = 1$. The address range of EPROM will be 0000_H to $7FFF_H$ and that of RAM will be 8000_H to $FFFF_H$.

Consider a system in which 32 kB memory space is implemented using four 8 kB memory. Let two 8 kB memory be EPROM and the remaining two be RAM. Each 8 kB memory requires 13 address lines. So, the address lines A_0 - A_{12} of the processor are connected to 13 address pins of all the memory ICs. The address lines A_{13} and A_{14} can be decoded using a 2-to-4 decoder to generate four chip select signals. These four chip select signals can be used to select one of the four memory IC at any one time. The address line A_{15} is used as an enable for the decoder. The simplified schematic of this memory organization is shown in Fig. 1.15 and address allotted to each memory IC is shown in Table 1.8.

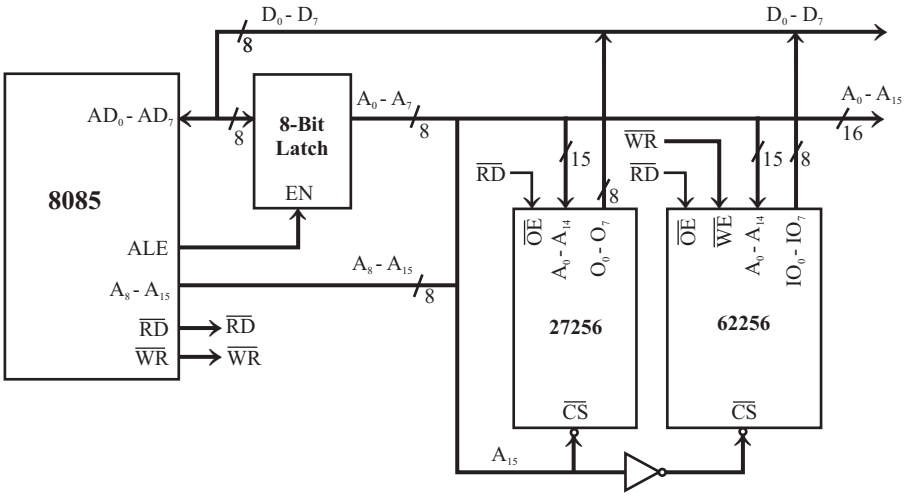


Fig. 1.14 : Example of implementing 32 kB EPROM and 32 kB RAM in an 8085 system.

Table 1.8: Address Allocation for Memory ICs Shown in Fig. 1.15

Device	Binary address																Hexa address
	Decoder enable/input				Input to address pins of memory IC												
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
8 kB EPROM - I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0001
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF
8 kB EPROM - II	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000
	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	2001
	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	2002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFF
8 kB RAM - I	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	4001
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	4002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	5FFF
8 kB RAM -II	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	6000
	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	6001
	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	6002
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7FFF

Consider a system in which the 64 kB memory space is implemented using eight numbers of 8 kB memory. Each 8 kB memory requires 13 address lines and so the address line A_0-A_{12} of the processor are connected to 13 address pins of all the memory ICs. The address lines A_{13}, A_{14} and A_{15} are decoded using a 3-to-8 decoder to generate eight chip select signals. These eight chip select signals can be used to select one of the eight memory IC at any one time. Design example-2 given at the end of this chapter is an example of implementing 64 kB address space using 8 numbers of 8 kB memory.

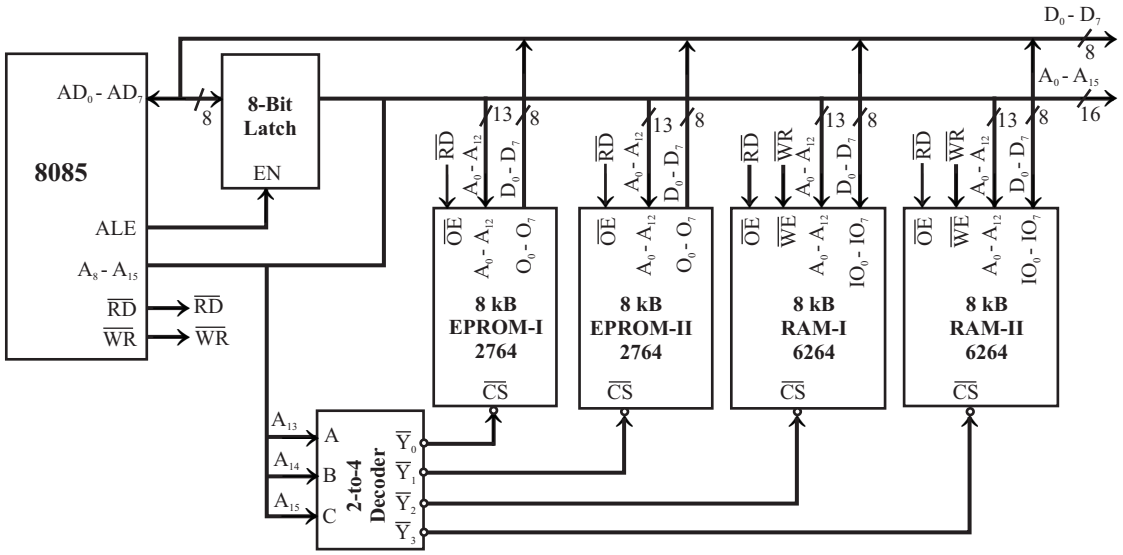


Fig. 1.15: Example of implementing 16 kB EPROM and 16 kB RAM in an 8085 system.

1.4 IO PORTS AND DATA TRANSFER CONCEPTS

The IO devices connected to a microcomputer system provides an efficient means of communication between the microcomputer system and the outside world. These IO devices are commonly called peripherals and include keyboards, CRT displays, printers and disks (floppy disk, hard disk and **C**ompact **D**isc (CD)).

The characteristics of the IO devices are normally different from the characteristics of the microprocessor. Since the characteristics of the IO devices are not compatible with that of the microprocessor, interface hardware circuitry between the microprocessor and IO device are necessary.

There are three major types of data transfer between the microcomputer and an IO device. They are as follows:

- **Programmed IO**
- **Interrrupt driven IO**
- **Direct memory access (DMA)**

(AU, Nov/Dec' 19, 2 Marks)

In programmed IO the data transfer is accomplished through an IO port and controlled by software. In interrupt driven IO, the IO device will interrupt the processor and initiate data transfer. In DMA, the data transfer between memory and IO can be performed by bypassing the microprocessor. Each type of data transfer scheme mentioned above, includes different methods of data transfer schemes. Fig. 1.16 shows all the types of data transfer schemes in a microcomputer and it can also be called **IO structure of a microcomputer**.

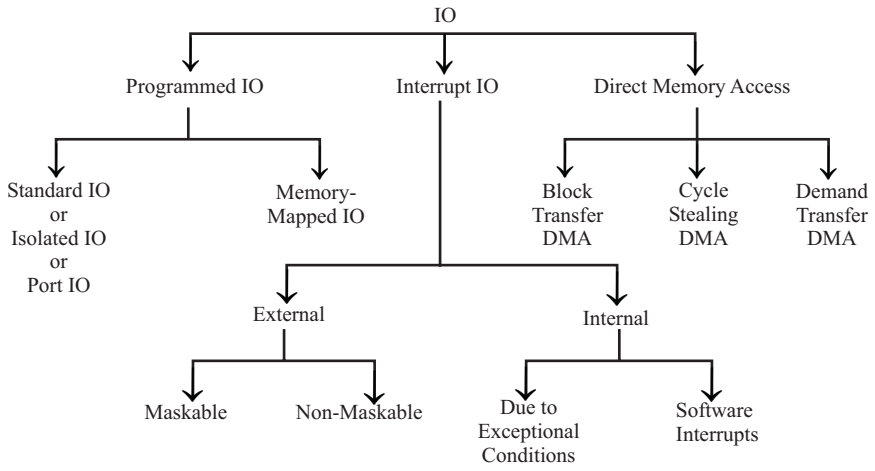


Fig. 1.16: IO structure of a typical microcomputer.

1.4.1 INTERFACING IO AND PERIPHERAL DEVICES

The IO devices are generally slow devices. So, they are connected to the system bus through ports. The ports are buffer IC which is used to temporarily hold the data transmitted from the microprocessor to IO device or to hold the data transmitted from IO device to the microprocessor.

To data transfer from the input device to the processor the following operations are performed:

- **The input device will load the data to the port.**
- **When the port receives the data, it sends message to the processor to read the data.**
- **The processor will read the data from the port.**
- **After the data has been read by the processor the input device will load the next data into the port.**

To data transfer from the processor to the output device the following operations are performed:

- **The processor will load the data to the port.**
- **The port will send a message to the output device to read the data.**
- **The output device will read the data from the port.**
- **After the data has been read by the output device the processor can load the next data to the port.**

INTEL IO Port Devices

The various INTEL IO port devices are 8212, 8155/8156, 8255, 8355 and 8755.

INTEL 8212

The 8212 is a 24-pin IC. It consists of eight number of D-type latches, each followed by a tristate buffer. It has 8-input lines DI_1 to DI_8 and 8-output lines DO_1 to DO_8. The 8212 can be used as an input or output device and the function is determined by the mode pin. However, it cannot be used simultaneously for input and output in the same circuit, since its mode pin is hardwired. It has 2-device select signals DS_1 and DS_2. The port is selected by the processor by sending appropriate address to device select pins.

Output Port : When MD = 1, $DS_1 = 0$ and $DS_2 = 1$

Input Port : When MD = 0, $DS_1 = 0$ and $DS_2 = 1$

INTEL 8155

INTEL 8155 has 256 × 8 static RAM, two numbers of 8-bit parallel IO port (ports A and B), one number of 6-bit parallel IO port (port-C) and 14-bit timer. The ports A and B can be programmed to work as simple or handshake input or output port. If port-A and port-B are simple ports then port-C can be used as input or output port. The timer can be programmed to operate in four different modes. INTEL 8155 requires six internal addresses and has one logic **low** **C**hip **S**elect pin (CS). The addresses of internal devices of 8155 are listed in Table 1.9.

Table 1.9: Internal Address of 8155/8156

Internal device	A_2	A_1	A_0
Control Register/ Status Register	0	0	0
Port-A	0	0	1
Port-B	0	1	0
Port-C	0	1	1
LSB of Timer	1	0	0
MSB of Timer	1	0	1

INTEL 8156

INTEL 8156 is same as 8155, but it has logic **high** **C**hip **S**elect (CS), i.e., the chip is selected when CS = 1.

INTEL 8255

It has 3 numbers of 8-bit parallel IO ports (ports A, B and C).

Port-A can be programmed in mode 0, mode1 or mode-2 as input or output port. Port-B can be programmed in mode-1 and mode-2 as IO port.When ports A and B are in mode-0, port-C can be used as IO port. The individual pins of port-C can be set or reset. INTEL 8255 requires four internal addresses and has one logic **low** **C**hip **S**elect (CS) pin. The address of internal devices of 8255 are listed in Table 1.10.

Table 1.10: Internal Address OF 8255

Internal device	A_1	A_0
Port-A	0	0
Port-B	0	1
Port-C	1	0
Control Register	1	1

INTEL 8355

It has 2 k × 8 ROM and two numbers of 8-bit port (Ports A and B). The individual pins of ports A and B can be programmed as input or output lines by sending a control word to DDR (**D**ata **D**irection **R**egister). The address of internal devices of 8355 are listed in Table 1.11. The 8355 requires four internal addresses and has one logic **low** **C**hip **S**elect (CS) pin.

INTEL 8755

Same as 8355 but has 2 k × 8 EPROM.

INTEL peripheral devices

Apart from port ICs, dedicated programmable controller/peripheral ICs are used in the system for various activities. Some of the controller/peripheral devices used in the 8085 system and their functions and internal addresses are listed in Table 1.11.

Table 1.11: Internal Address of 8355/8755

Internal device	A_1	A_0
Port-A	0	0
Port-B	0	1
DDR A	1	0
DDR B	1	1

Table 1.12: Functions and Internal Addresses of Peripheral Devices

Device	Function	Internal addresses
INTEL 8279	Keyboard/display controller. Used for keyboard scanning and display refreshing.	**Two-internal addresses** $A_0 = 0 \rightarrow$ Data register $A_0 = 1 \rightarrow$ Control register
INTEL 8257 or INTEL 8237	DMA controller. Used for supporting DMA access to the IO device. It acts as a master during the DMA mode. It is a slave device during programming mode.	**Sixteen-internal addresses** A_3 A_2 A_1 A_0 0 0 0 0 0 0 0 1 . . . . 1 1 1 1
INTEL 8259	Interrupt controller. Used to expand the hardware interrupt INTR to eight interrupts in an 8085-based system and 256 interrupts in an 8086-based system.	**Two-internal addresses** $A_0 = 0$ $A_0 = 1$
INTEL 8253/ 8254	Programmable timer. Used in the system to produce various timing signals. It has three independent counters and can be programmed in six operating modes.	**Four-internal addresses** A_1 A_0 Counter-0 0 0 Counter-1 0 1 Counter-2 1 0 Control Register 1 1
INTEL 8251 (USART)	**U**niversal **S**ynchronous/**A**synchronous **R**eceiver **T**ransmitter. Used for serial data communication.	**Two-internal addresses** C/D = 0 $\rightarrow$ Data register C/D = 1 $\rightarrow$ Control register

IO Mapping

The port and peripheral devices will have one logic **low**/**high** chip select pin. The processor can access the port/peripheral device by supplying internal address and chip select signals. Therefore, the port and peripheral device interfacing (IO interfacing) deals with allocation of various internal addresses and generation of chip select signals.

There are two ways of interfacing IO devices in 8085-based system.

- **Memory-mapped IO device.**
- **Standard IO-mapped IO device or Isolated IO mapping.**

Note: The interfacing of IO ports and controller/peripheral ICs are commonly referred as IO device mapping.

In memory mapping of IO devices the ports are allotted a 16-bit address like that of the memory location. Some of the chip select signals generated to select memory ICs are used for selecting the IO port devices. Hence, the processor treats the IO ports as memory locations for reading and writing (i.e., the devices which are mapped by memory mapping are accessed by executing memory read cycle or memory write cycle).

In standard IO mapping or isolated IO mapping, a separate 8-bit address is allotted for the IO ports and the peripheral ICs. The processor differentiates the IO-mapped devices, from the memory-mapped devices in the following ways:

1. **For accessing the IO-mapped devices the processor executes IO read or write cycle.**
2. **During IO read or write cycle, the 8-bit address is placed on both low order address lines and the high order address lines.**
3. **IO/$\overline{M}$ is asserted high to indicate the IO operation (for read as well as write).**

A 8085 processor does not provide separate read ($\overline{RD}$) and write ($\overline{WR}$) signals for memory and IO devices. But it differentiates the memory and IO device accessed by IO/$\overline{M}$ signal. The three signals $\overline{RD}$, $\overline{WR}$ and IO/$\overline{M}$ can be decoded as shown in Fig. 1.17 to provide separate read and write control signals for IO devices and memory devices.

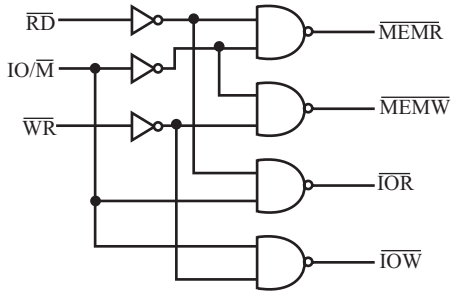


Fig. 1.17: Circuit to generate separate read and write signals for memory and IO devices in an 8085-based system.

When the devices are IO-mapped, then only IN and OUT instructions have to be used for data transfer between the device and the processor. For the IO-mapped devices a separate decoder should be used to generate the required chip select signals.

Table 1.13: Comparison of Memory Mapping and IO Mapping of IO Device

Memory mapping of IO device	IO mapping of IO device
1. 16-bit addresses are provided for IO devices.	1. 8-bit addresses are provided for IO devices.
2. The devices are accessed by memory read or memory write cycles.	2. The devices are accessed by IO read or IO write cycle. During these cycles, the 8-bit address is available on both low order address lines and high order address lines.
3. The IO ports or peripherals can be treated like memory locations and so all instructions related to memory can be used for data transfer between the IO device and the processor.	3. Only IN and OUT instructions can be used for data transfer between the IO device and the processor.
4. In memory-mapped ports, the data can be moved from any register to the ports and vice versa.	4. In IO-mapped ports, the data transfer can take place only between the accumulator and the ports.
5. When memory mapping is used for IO devices, the full memory address space cannot be used for addressing memory. Hence memory mapping is useful only for small systems, where the memory requirement is less.	5. When IO mapping is used for IO devices, then the full memory address space can be used for addressing the memory. Hence it is suitable for systems which requires a large memory capacity.
6. In memory-mapped IO devices, a large number of IO ports can be interfaced.	6. In IO mapping, only 256 ports (2^8 = 256) can be interfaced.
7. For accessing memory-mapped devices,the processor executes the memory read or write cycle. During this cycle, IO/$\overline{M}$ is asserted **low**(IO/$\overline{M}$= 0)	7. For accessing the IO-mapped devices, the processor executes the IO read or write cycle. During this cycle, IO/$\overline{M}$ is asserted **high** (IO/$\overline{M}$=1).

DESIGN EXAMPLE - 1

Interface two numbers of 4 kB EPROM and one number of 8 kB RAM with 8085 processor. Explain the interface diagram and allocate binary addresses to memory ICs.

Solution

The IC 2732 is selected for EPROM memory and the IC 6264 is selected for RAM memory. Both the memory IC's have time compatibility with 8085 processor.

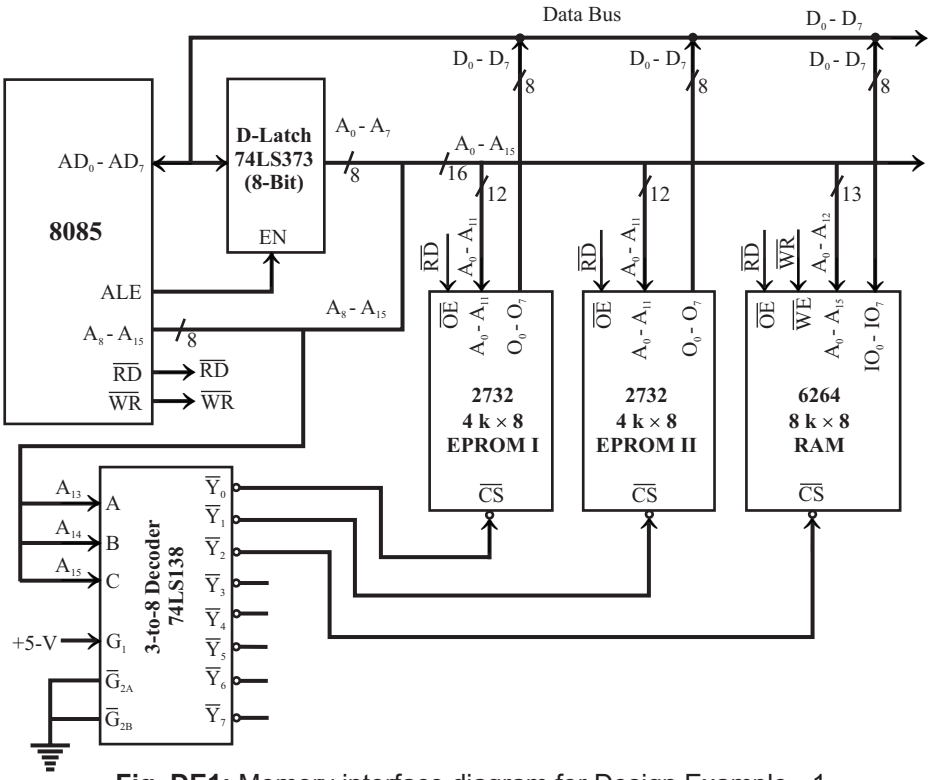


Fig. DE1: Memory interface diagram for Design Example - 1.

The 4 *kB* EPROM IC requires 12 address lines (2^{12} = 4 k). The 8 kB RAM IC requires 13 address lines (2^{13} = 8 k). The address lines A_0 - A_{11} are connected to both EPROM and RAM address input pins. The address lines A_{13}, A_{14} and A_{15} are not used for memory address. Hence by decoding these address lines we can generate chip select signals.

The 3-to-8 decoder, 74LS138 is employed to produce the chip select signals for the system. The decoder has 8-output lines which can be used as 8-chip select signals. In this, three chip select signals are used for selecting memory ICs and the remaining five can be used for selecting other peripheral ICs in the system or for future expansion of the memory capacity. The interface diagram is shown in Fig. DE1. Address allotted to memory ICs are shown in Table DE1.

The EPROM's are mapped in the beginning of memory space. The remaining addresses can be allotted to RAM's. The EPROM memory is mapped from 0000_H to $0FFF_H$ and 2000_H to $2FFF_H$. The RAM memory is mapped from 4000_H to $5FFF_H$.

Table DE1: Address Allocation Table for Design Example - 1

Memory IC	Binary address																Hexa address
	Decoder input			Input to memory address pins													
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
EPROM I 2732	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0000
	0	0	0	X	0	0	0	0	0	0	0	0	0	0	0	1	0001
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	0	X	1	1	1	1	1	1	1	1	1	1	1	1	0FFF
EPROM II 2732	0	0	1	X	0	0	0	0	0	0	0	0	0	0	0	0	2000
	0	0	1	X	0	0	0	0	0	0	0	0	0	0	0	1	2001
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	1	X	1	1	1	1	1	1	1	1	1	1	1	1	2FFF
RAM 6264	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	4001
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	5FFF

Note: X indicates the unused address line for the particular memory IC and they are considered as zero.

DESIGN EXAMPLE - 2

Interface three numbers of 8 kB EPROM and 5 numbers of 8 kB static RAM to microprocessor 8085 to have a total memory capacity of 64 kB.

Solution

The IC 2764 is selected for EPROM memory and the IC 6264 is selected for RAM memory. Both the memory ICs have time compatibility with the 8085 processor.

The 8 kB EPROM IC requires 13 address lines (2^{13} = 8 k). The 8 kB RAM IC also requires 13 address lines (2^{13} = 8 k). The address lines A_0 - A_{12} are connected to all the EPROM's and RAMs. Hence A_0 - A_{12} will select the required memory location.The address lines A_{13}, A_{14} and A_{15} are not used for memory address. Hence by decoding these address lines we can generate chip select signals.

The 3-to-8 decoder, 74LS138 is employed to produce the chip select signals for the system. The decoder has 8-output lines which can be used as 8-chip select signals. All the 8-chip select signals are used to select memory ICs. EPROM's are mapped at the beginning of memory space. The decoder will select a memory IC by decoding the address lines A_{13}, A_{14} and A_{15}. The address lines A_0 - A_{12} will select a particular memory location in the selected IC. The interface diagram is shown in Fig. DE2 and address allocation table is shown in Table DE2.

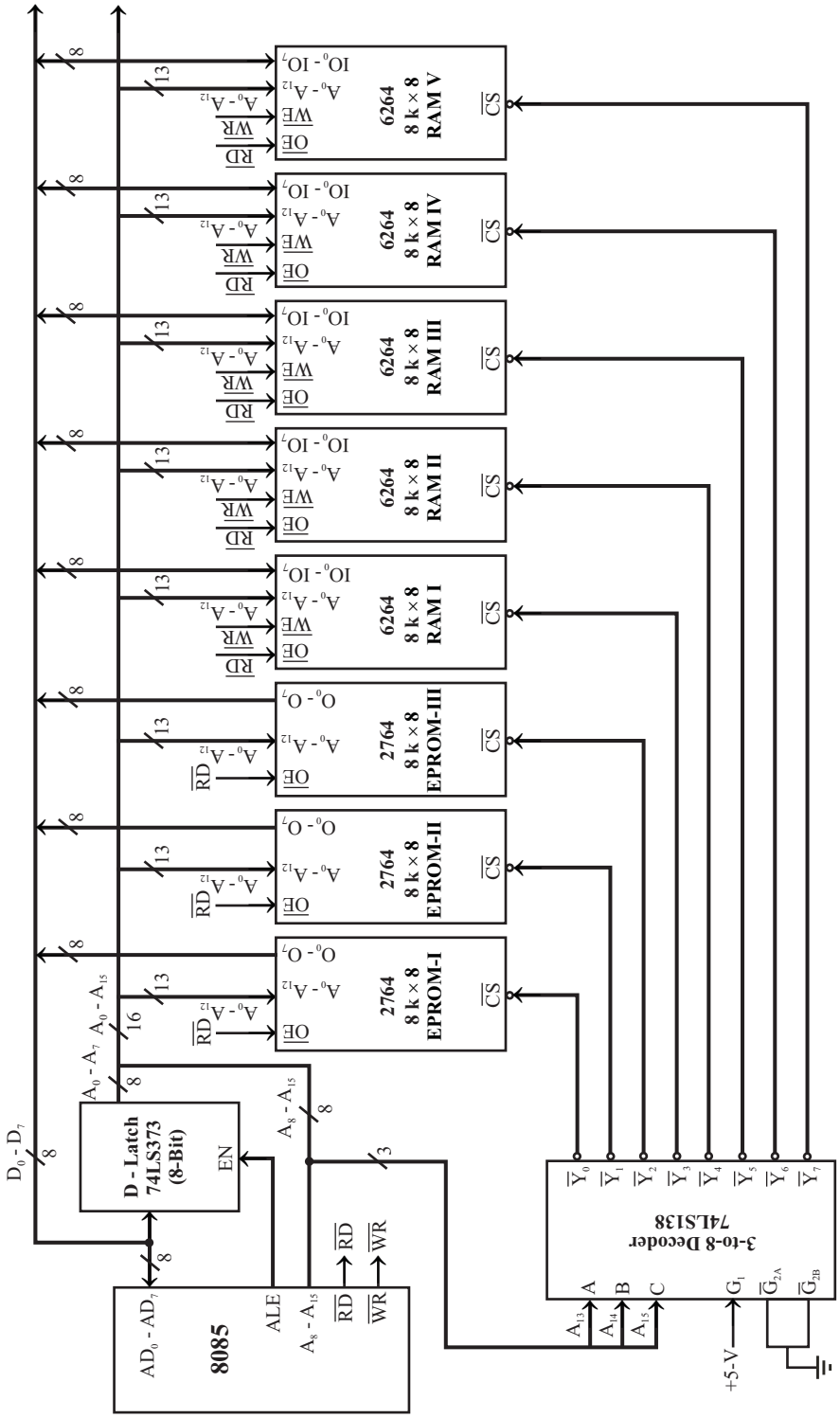


Fig. DE2: Memory interface diagram for Design Example - 2.

Table DE2: Address Allocation Table for Design Example- 2

Memory IC chip	Binary address																Hexa address
	Decoder input			Input to memory address pins													
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
EPROM I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0001
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF
EPROM II	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000
	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	2001
	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	2002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFF
EPROM III	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4000
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	4001
	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	4002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	5FFF
RAM I	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	6000
	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	6001
	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	6002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7FFF
RAM II	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	8001
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	8002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	9FFF
RAM III	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	A000
	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	A001
	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	A002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	BFFF
RAM IV	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C000
	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	C001
	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	C002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	DFFF
RAM V	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	E000
	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	E001
	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	E002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFF

In this system the full memory capacity of 64 *kB* is utilized for memory. Hence the peripheral ICs and the IO ports should be IO-mapped in the system. The EPROM is mapped from 0000_H to $5FFF_H$. The RAM is mapped from 6000_H to $FFFF_H$. The EPROM capacity is 24 *kB*. The RAM capacity is 4 *kB*.

DESIGN EXAMPLE – 3

In a microprocessor system using 8085, the memory requirement is 8 kB EPROM and 8 kB RAM. For interfacing IO devices, three numbers of 8255 are required. Select suitable memories and explain how they are interfaced to the system. Interface the 8255 by memory mapping.

Solution

The IC 2764 is selected for EPROM memory and the IC 6264 is selected for RAM memory. Both the memory IC's have time compatibility with 8085 processor.

The 8 kB EPROM, 2764 requires 13 address lines (2^{13} = 8 k). The 8 kB RAM, 6264 also requires 13 address lines (2^{13} = 8 k). The address lines A_0 to A_{12} are connected to both EPROM and RAM memory ICs. The 8255 requires four internal addresses. Let us connect A_1 of 8085 to A_0 of 8255 and A_2 of 8085 to A_1 of 8255. The 8255 is memory-mapped in the system.

> *Note: The internal devices of 8255 can be selected by connecting any two address lines of the processor to A_0 and A_1 of 8255.*

For the memories and 8255's we require 5 chip select signals. Hence we can use a 3-to-8 decoder 74LS138 for generating eight chip select signals by decoding the unused address lines A_{13}, A_{14} and A_{15}. The decoder enabled pins are permanently tied to appropriate levels. In the eight chip select signals, five are used for selecting memory ICs and 8255 and the remaining three can be used for future expansion. The memory/8255 interface diagram is shown in Fig. DE3.

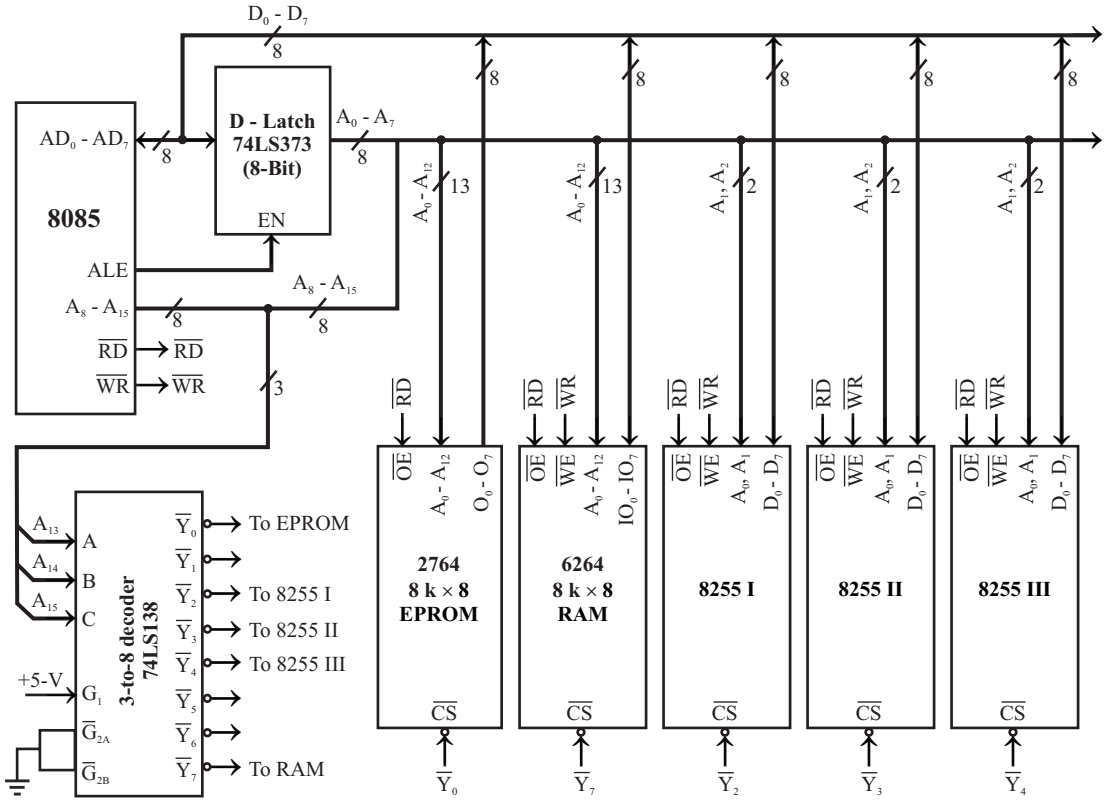


Fig. DE3: Memory interface diagram for Design Example - 3.

Table DE3: Address Allocation Table for Design Example - 3

Device	Binary address																Hexa
	Decoder input			Input to address pins of memory/8255													
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	address
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
2764	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0001
EPROM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0002
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF
	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	E000
	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	E001
6264	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	E002
RAM	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFF
8255 I																	
Port-A	0	1	0	X	X	X	X	X	X	X	X	X	X	0	0	X	4000
Port-B	0	1	0	X	X	X	X	X	X	X	X	X	X	0	1	X	4002
Port-C	0	1	0	X	X	X	X	X	X	X	X	X	X	1	0	X	4004
Control register	0	1	0	X	X	X	X	X	X	X	X	X	X	1	1	X	4006
8255 II																	
Port-A	0	1	1	X	X	X	X	X	X	X	X	X	X	0	0	X	6000
Port-B	0	1	1	X	X	X	X	X	X	X	X	X	X	0	1	X	6002
Port-C	0	1	1	X	X	X	X	X	X	X	X	X	X	1	0	X	6004
Control register	0	1	1	X	X	X	X	X	X	X	X	X	X	1	1	X	6006
8255 III																	
Port-A	1	0	0	X	X	X	X	X	X	X	X	X	X	0	0	X	8000
Port-B	1	0	0	X	X	X	X	X	X	X	X	X	X	0	1	X	8002
Port-C	1	0	0	X	X	X	X	X	X	X	X	X	X	1	0	X	8004
Control register	1	0	0	X	X	X	X	X	X	X	X	X	X	1	1	X	8006

Note: The X indicates that the address line is not used for the particular device and they are considered as zero.

The EPROM is mapped at the starting of memory space. The RAM is mapped at the end of memory space. The EPROM is mapped from 0000_H to $1FFF_H$. The RAM is mapped from $E000_H$ to $FFFF_H$. The four internal devices of 8255 are control register, port-A, port-B and port-C. A 16-bit address is allotted to each internal device of 8255 as shown in Table-DE3.

DESIGN EXAMPLE - 4

Interface 2 kB RAM and 256 × 8 ROM with 8085 processor to satisfy the total memory requirement of 8 kB RAM and 1 kB ROM.

Solution

The memory requirement of 8 kB RAM can be achieved with 4 numbers of 2 kB RAM. The memory requirement of 1kB ROM can be achieved with 4 numbers of 256 × 8 ROM. (4 × 256 = 1024 = 1k). The 2 kB RAM requires 11 address lines (2^{11} = 2 k). The 256 × 8 ROM requires 8 address lines (2^8 = 256).

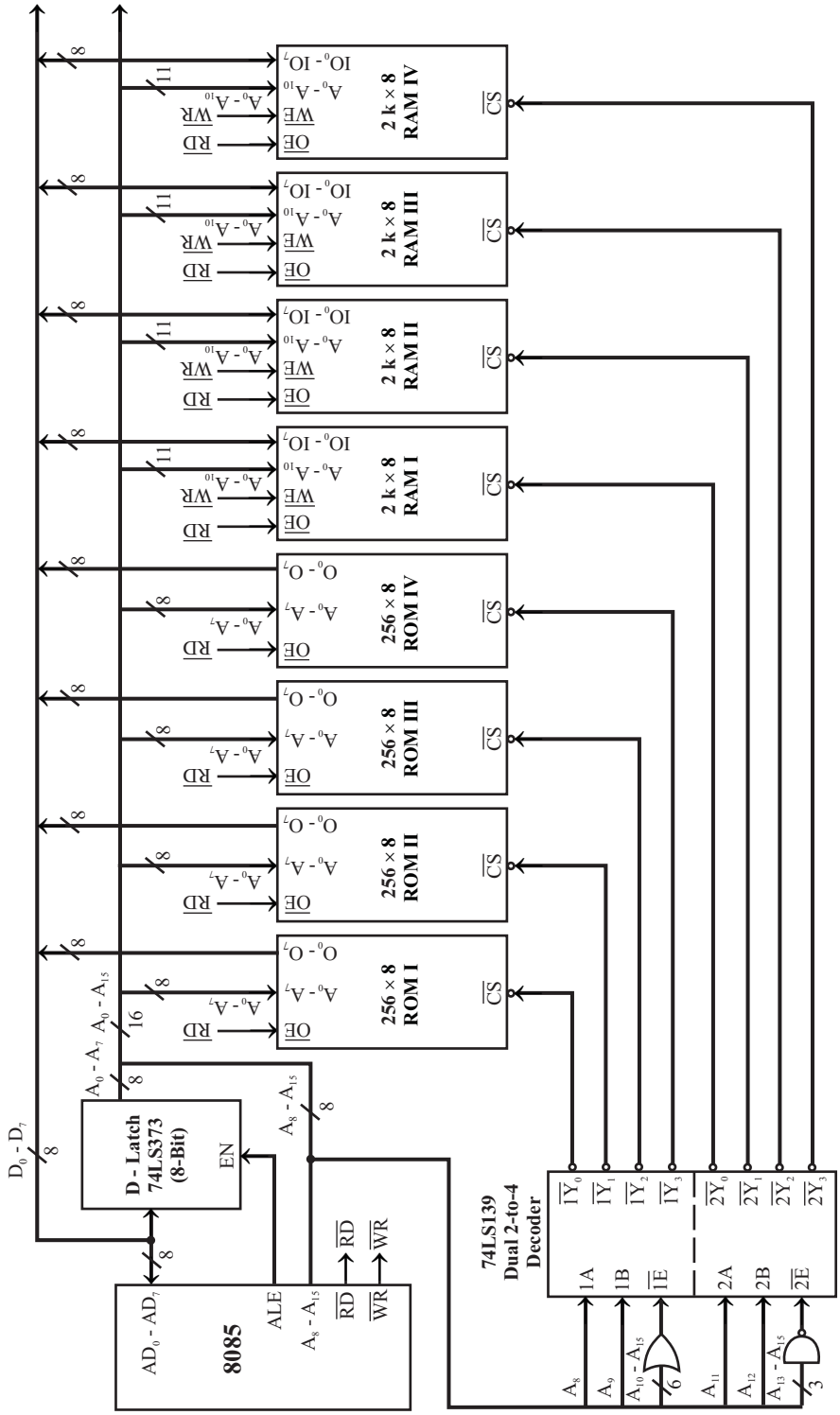


Fig. DE4: Memory interface diagram for Design Example - 4.

Table DE4: Address Allocation Table for Design Example - 4

Device	Binary address																Hexa address
	ROM decoder enable						Decoder input		Input to ROM memory address pins								
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
256 × 8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0001
ROM I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0002
		.			.		.			.				.			.
		.			.		.			.				.			.
	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	00FF
	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0100
256 × 8	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0101
ROM II	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	0	0102
		.			.		.			.				.			.
		.			.		.			.				.			.
	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	01FF
	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0200
256 × 8	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0201
ROM III	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0202
		.			.		.			.				.			.
		.			.		.			.				.			.
	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	02FF
	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0300
256 × 8	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	0301
ROM IV	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0302
		.			.		.			.				.			.
		.			.		.			.				.			.
	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	03FF

Device	Binary address																Hexa address
	RAM decoder enable			Decoder input		Input to RAM memory address pins											
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	E000
2 k × 8	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	E001
RAM I	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	E002
	.		.			.			.				.				.
	.		.			.			.				.				.
	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	E7FF
	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	E800
2 k × 8	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	1	E801
RAM II	1	1	1	0	1	0	0	0	0	0	0	0	0	0	1	0	E802
	.		.			.			.				.				.
	.		.			.			.				.				.
	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	EFFF
	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	F000
2 k × 8	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	F001
RAM III	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	F002
	.		.			.			.				.				.
	.		.			.			.				.				.
	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	F7FF
	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	F800
2 k × 8	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	F801
RAM IV	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	0	F802
	.		.			.			.				.				.
	.		.			.			.				.				.
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFF

The address lines A_0 - A_{10} are connected to RAM ICs. Hence, they will select the required memory location in that ICs. The address lines A_0 - A_7 are connected to ROM ICs. Hence, they will select the required memory location in those ICs.

Totally there are 8-memory ICs hence we require 8-chip select signals. The 8-chip select signals can be generated by using a dual 2-to-4 decoder 74LS139. One of the 2-to-4 decoder is used to generate chip select signals for ROM memory ICs and the other decoder is used to generate chip select signals for RAM memory ICs. The address lines A_8 and A_9 are used to generate chip select signals for ROM memory. The address lines A_{10} to A_{15} are logically ORed and used as enable for ROM decoder. The address lines A_{11} and A_{12} are used to generate chip select signals for RAM memory. The address lines A_{13} to A_{15} are logically NANDed and used as enable for RAM decoder.

ROM memories are mapped in the beginning of memory space. The RAM memories are mapped at the end of memory space. The ROM memories are mapped from 0000_H to $03FF_H$. The RAM memories are mapped from $E000_H$ to $FFFF_H$.

DESIGN EXAMPLE – 5

A system requires 16 kB EPROM and 16 kB RAM. Also the system has 2 numbers of 8255, one number of 8279, one number of 8251 and one number of 8254.

(8255 - Programmable peripheral interface, 8279-Keyboard/display controller, 8251 - USART and 8254 - Timer)

Draw the Interface diagram. Allocate addresses to all the devices. The peripheral IC's should be IO-mapped.

Solution

The IO devices in the system should be mapped by standard IO mapping. Hence separate decoders can be used to generate chip select signals for memory IC's and peripheral IC's.

For 16 *kB* EPROM, we can provide 2 numbers of 2764 (8 k × 8) EPROM. For 16 *kB* RAM we can provide 2 numbers of 6264 (8 k × 8) RAM.

The 8 *kB* memory requires 13 address lines (2^{13} = 8 k). Hence the address lines A_0 - A_{12} are used for selecting the memory locations. The unused address lines A_{13}, A_{14} and A_{15} are used as input to decoder 74LS138 (3-to-8-decoder) of memory IC. The logic **low** enables of this decoder are tied to IO/M of 8085, so that this decoder is enabled for memory read/write operation. The other enable pins of decoder are tied to appropriate logic levels permanently. The 4 outputs of the decoder are used to select memory IC's and the remaining 4 are kept for future expansion.

The EPROM is mapped in the beginning of memory space from 0000_H to $3FFF_H$. The RAM is mapped at the end of memory space from $C000_H$ to $FFFF_H$.

There are five peripheral IC's to be interfaced to the system. The chip select signals for these IC's are given through another 3-to-8 decoder 74LS138 (IO decoder). The input to this decoder is A_{10}, A_{11} and A_{12}. The address lines A_{13}, A_{14} and A_{15} are logically ORed and applied to **low** enable of IO decoder. The logic **high** enable of IO decoder is tied to IO/$\overline{M}$ signal of 8085, so that this decoder is enabled for IO read/write operation.

Here, the high order address lines can be used for decoding because the processor outputs the 8-bit port address both on AD_0 to AD_7 and A_8 to A_{15}. The address lines A_0 and A_1 are used to select the internal devices of the peripheral ICs. The output of the decoder are used to select the ICs. Three outputs of the decoder will be spare for future expansion.

Note:	*Since the IO devices are IO-mapped in the system, 8-bit addresses have been allotted to them.*

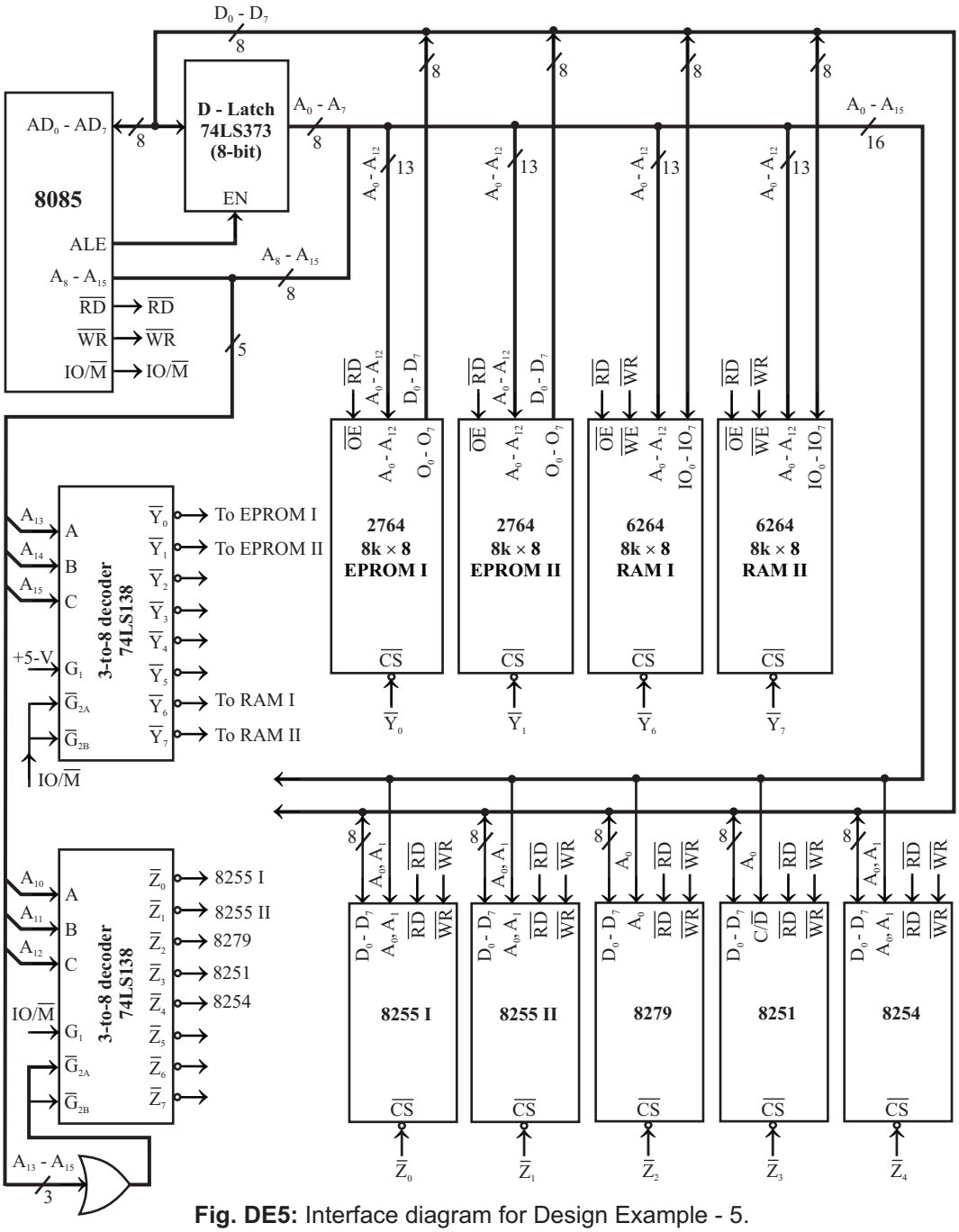


Fig. DE5: Interface diagram for Design Example - 5.

Table DE5: Address Allocation Table for Design Example - 5

Device	Binary address																Hexa address
	Input to memory decoder			Input to memory address pins													
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
2764 I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
8 k × 8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF
2764 II	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000
8 k × 8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	3FFF
6264 I	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C000
8 k × 8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	DFFF
6264 II	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	E000
8 k × 8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	FFFF

Device	Binary address								Hexa address
	IO decoder enable			IO decoder input			Input to IO device address pins		
	A_{15} A_7	A_{14} A_6	A_{13} A_5	A_{12} A_4	A_{11} A_3	A_{10} A_2	A_9 A_1	A_8 A_0	
8255 I									
Port-A	0	0	0	0	0	0	0	0	00
Port-B	0	0	0	0	0	0	0	1	01
Port-C	0	0	0	0	0	0	1	0	02
Control register	0	0	0	0	0	0	1	1	03
8255 II									
Port-A	0	0	0	0	0	1	0	0	04
Port-B	0	0	0	0	0	1	0	1	05
Port-C	0	0	0	0	0	1	1	0	06
Control register	0	0	0	0	0	1	1	1	07
8279									
Data register	0	0	0	0	1	0	X	0	08
Control register	0	0	0	0	1	0	X	1	09
8251									
Data register	0	0	0	0	1	1	X	0	0C
Control register	0	0	0	0	1	1	X	1	0D
8254									
Counter-0	0	0	0	1	0	0	0	0	10
Counter-1	0	0	0	1	0	0	0	1	11
Counter-2	0	0	0	1	0	0	1	0	12
Control register	0	0	0	1	0	0	1	1	13

Note: *Don't care (X) is considered as zero.*

DESIGN EXAMPLE – 6

In a microprocessor-based system 8085, 8 kB EPROM and 8 kB RAM are needed. For interfacing IO devices two numbers of 8155 are required. Select suitable memories and explain how they are interfaced in the system. Interface the 8155 ports by IO mapping.

Solution

The IC 2764 (8 k × 8) is selected for EPROM memory and IC 6264 (8 k × 8) is selected for RAM memory. Both the memory IC's have time compatibility with 8085 processor.

The 8 *kB* memories require 13 address lines (2^{13} = 8 k). Hence, the address lines A_0 - A_{12} are used to select memory locations.

In addition to 6264, each one of the 8155 chip provides a static RAM capacity of 256 bytes. The RAM locations of 8155 are selected by address lines A_0-A_6.

A 3-to-8 decoder, 74LS138 is used for generating chip select signals by decoding the address lines A_{13}, A_{14} and A_{15}.

The 8155 has internal address latch and decoder to differentiate memory operation and IO operation. To utilize this facility, the control signals ALE and IO/M are connected to 8155.

The 8155 ports and memory locations can be selected from the decoder used for memory devices. It differentiates the memory and IO operation from IO/M signal. Eight bit addresses are allotted to ports of 8155 and sixteen bit addresses are allotted to RAM memory locations of 8155.

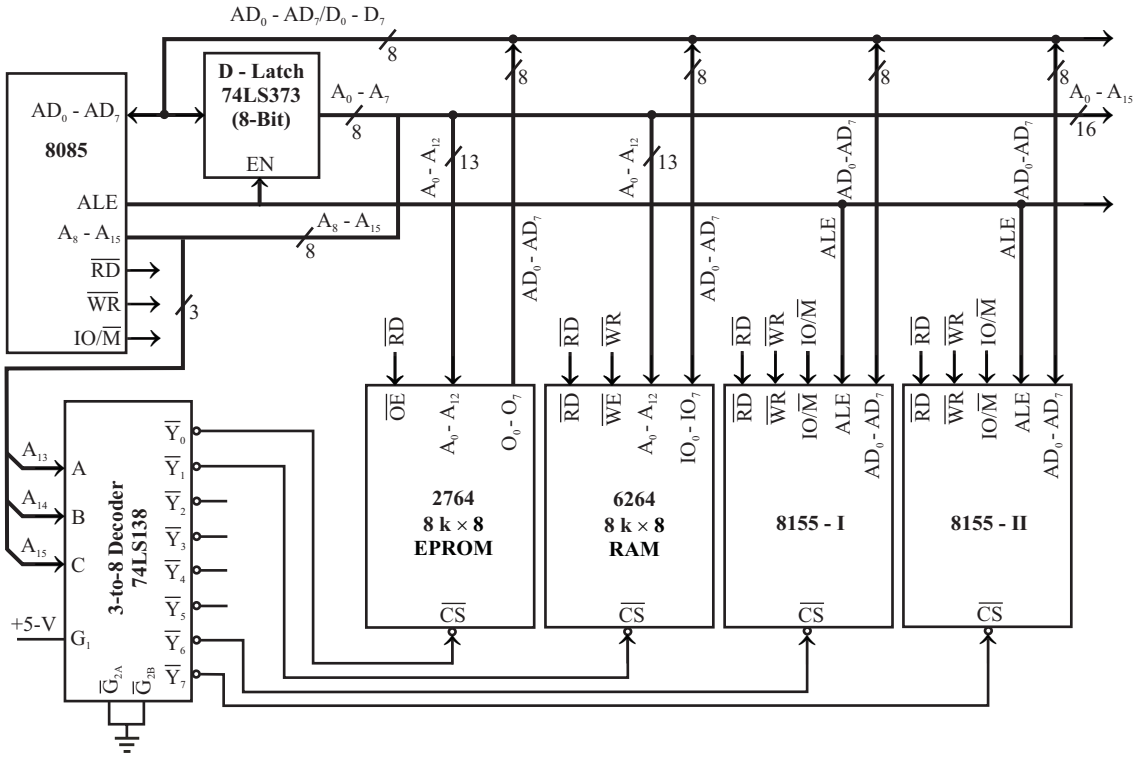


Fig. DE6: Interface diagram for Design Example 6.

Table DE6: Address Allocation Table for Design Example - 6

Device	Binary address																Hexa address
	Decoder input			Input to address pins of memory/8155													
	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0000
2764	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0001
8 k × 8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
EPROM	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1FFF
	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2000
	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	2001
6264	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
8 k × 8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
RAM	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	3FFF
8155 I																	
RAM	1	1	0	X	X	X	X	X	0	0	0	0	0	0	0	0	C000
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
256 × 8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	1	1	0	X	X	X	X	X	1	1	1	1	1	1	1	1	C0FF
Control register	1	1	0	X	X	0	0	0									C0
Port-A	1	1	0	X	X	0	0	1									C1
Port-B	1	1	0	X	X	0	1	0									C2
Port-C	1	1	0	X	X	0	1	1									C3
LSB timer	1	1	0	X	X	1	0	0									C4
MSB timer	1	1	0	X	X	1	0	1									C5
8155 II																	
RAM	1	1	1	X	X	X	X	X	0	0	0	0	0	0	0	0	E000
	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
256 × 8	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
	1	1	1	X	X	X	X	X	1	1	1	1	1	1	1	1	E0FF
Control register	1	1	1	X	X	0	0	0									E0
Port-A	1	1	1	X	X	0	0	1									E1
Port-B	1	1	1	X	X	0	1	0									E2
Port-C	1	1	1	X	X	0	1	1									E3
LSB timer	1	1	1	X	X	1	0	0									E4
MSB timer	1	1	1	X	X	1	0	1									E5

Note: *Don't care (X) is considered as zero.*

1.4.2 PARALLEL DATA COMMUNICATION INTERFACE

In microprocessor-based systems, digital information can be transmitted from one system to another system either by parallel or serial data transfer scheme.

In parallel data transfer, a group of bits (for eg., 8 bits) is transmitted from one device to another at any one time. To achieve parallel data transfer scheme, a group of data lines will be connecting the processor and peripheral devices. Normally in microprocessor-based systems the parallel data transfer schemes are adopted to transfer data between various devices inside the system.

Basically the microprocessor-based system has been fabricated on a PCB (**P**rinted **C**ircuit **B**oard) in which a bus is formed with the required number of data lines and the bus connects all the devices in the system. The data transmitted over the bus in a PCB are highly reliable. In a well designed board, there will not be any loss of data and the data will not be corrupted.

When data has to be transmitted over longer distances (i.e., greater than 0.5m), we require high current signals to drive the data for longer distance. In such cases data is transmitted bit by bit through a single data line.

Parallel Data Transfer Schemes

Data transfer schemes refer to the method of data transfer between the processor and peripheral devices. In a typical microcomputer, data transfer takes place between any two devices: microprocessor and memory, microprocessor and IO devices, or memory and IO devices. For effective data transfer between these devices, the timing parameters of the devices should be matched. But most of the devices have incompatible timings. For example, an IO device may be slower than the processor due to which it cannot send data to the processor at the expected time.

Semiconductor memories are available with compatible timings. Moreover, slow memories can be interfaced using additional hardware to introduce wait states in machine cycles. The microprocessor system designer often faces difficulties while interfacing IO devices and magnetic memories (like floppy or hard disk) to achieve efficient data transfer to or from the microprocessor. Several data transfer schemes have been developed to solve the interfacing problems with IO devices.

The data transfer schemes have been broadly classified into the following two categories :

1. **Programmed data transfer.**
2. **Direct memory access (DMA) data transfer.**

In programmed data transfer, a memory resident routine (subroutine) requests the device for data transfer to or from one of the processor register.

Programmed data transfer scheme is used when relatively small amount of data is to be transferred. In these schemes, usually one byte or word of data is transferred at a time. Examples of devices using programmed data transfer are ADC, DAC, Hex-keyboard, 7-segment LEDs, etc.

Programmed data transfer scheme can be further classified into the following three types:

a) **Synchronous data transfer scheme.**

b) **Asynchronous data transfer scheme.**

c) **Interrupt driven data transfer scheme.**

In DMA data transfer, the processor is forced to **HOLD** state (**high impedance** state) by an IO device until the data transfer between the device and the memory is complete. The processor does not execute any instruction during the **HOLD** period.

The DMA data transfer is used for large block of data transfer between IO device and memory. Typical examples of devices using DMA are CRT controller, floppy disk, hard disk, high speed line printer, etc.

The different types of DMA data transfer schemes are as follows:

a) **Cycle stealing DMA or Single transfer mode DMA.**

b) **Block or Burst mode DMA.**

c) **Demand transfer mode DMA.**

Figure 1.18 shows the various types of data transfer schemes. All the data transfer schemes discussed above requires both software and hardware for their implementation. Within a microcomputer, more than one scheme can be used for interfacing different IO devices. However, some of these schemes require specific hardware features in the microprocessor for implementing the scheme.

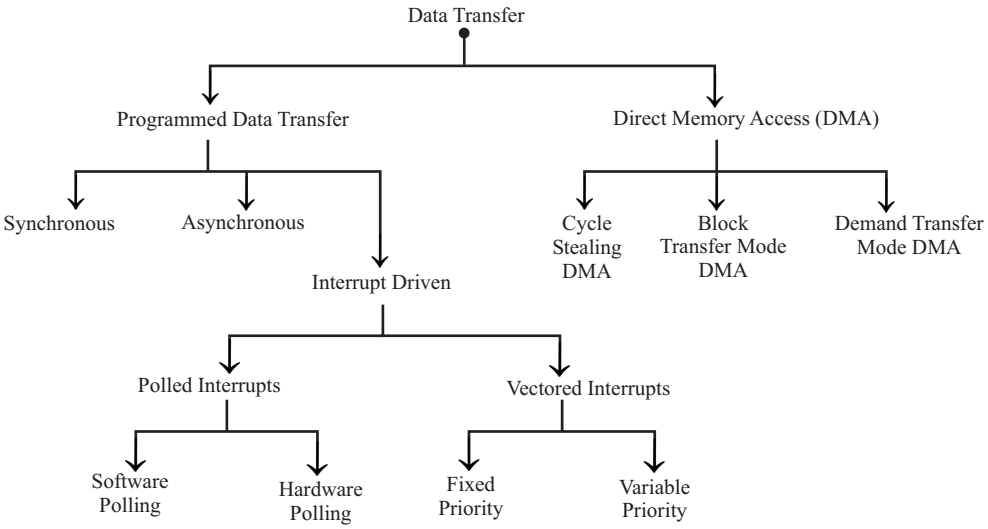


Fig. 1.18: Types of data transfer schemes.

Synchronous Data Transfer Scheme

The synchronous data transfer scheme is the simplest of all data transfer schemes. In this scheme the processor does not check the readiness of the device. The IO device or peripheral should have matched timing parameters. Whenever data is to be obtained from the device or transferred to the

device, the user program can issue a suitable instruction for the device. At the end of the execution of this instruction, the transfer would have been completed.

The synchronous data transfer scheme can also be implemented with a small delay (if the delay is tolerable) after the request has been made. The sequence of operations for synchronous data transfer scheme is shown in Fig. 1.19. The mode-0 input/output in 8255 is an example of synchronous data transfer.

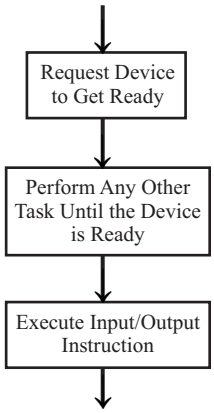


Fig. 1.19: Synchronous data transfer scheme.

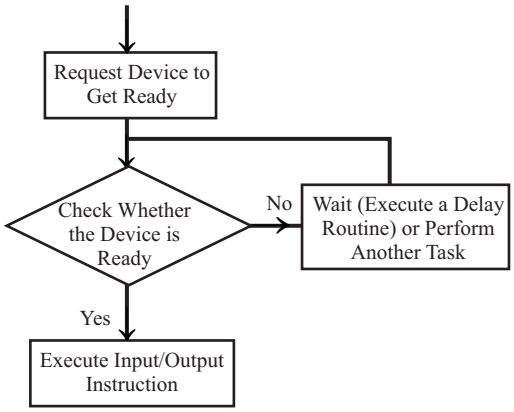


Fig. 1.20: Asynchronous data transfer scheme.

Asynchronous Data Transfer Scheme

The asynchronous data transfer scheme is employed when the speed of the processor and IO device does not match. In this scheme the processor sends a request to the device for read/write operation. Then the processor keeps on polling the status of the device. Once the device is ready, the processor executes a data transfer instruction to complete the process. To implement this scheme, the device should provide a signal which may be tested by the processor to ascertain whether it is ready or not.

The sequence of operations for asynchronous data transfer is shown in Fig. 1.20. The mode-1 and mode-2 handshake data transfer of 8255 without interrupt is an example of asynchronous data transfer.

Interrupt Driven Data Transfer Scheme

The interrupt driven data transfer scheme is the best method of data transfer for efficient utilization of processor time. In this scheme, the processor first initiates the IO device for data transfer. After initiating the device, the processor will continue the execution of instructions in the program. Also, at the end of every instruction the processor will check for a valid interrupt signal. If there is no interrupt then the processor will continue the execution.

When the IO device is ready, it will interrupt the processor. On receiving an interrupt signal the processor will complete the current instruction execution and save the processor status in stack. Then

the processor calls an **I**nterrupt **S**ervice **R**outine (ISR) to service the interrupting device. At the end of ISR, the processor status is retrieved from the stack and the processor starts executing its main program. The sequence of operations for an interrupt driven data transfer scheme is shown in Fig. 1.21.

Note: The user/system designer need not write any subroutine/procedure to check for an interrupt. The logic of checking interrupt signals while executing each instruction is incorporated in the processor itself by the manufacturer of the processor.

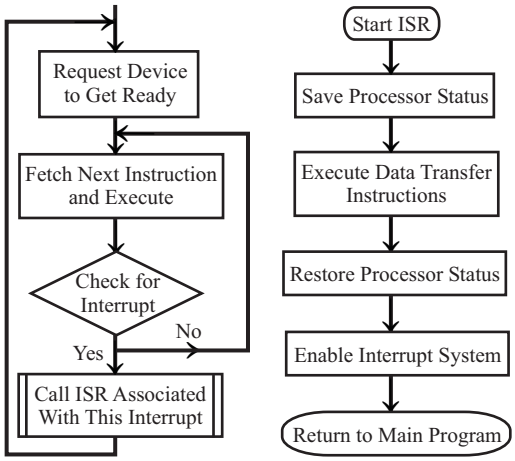


Fig. a: Main program execution sequence. **Fig. b:** ISR execution sequence.

Fig. 1.21: Interrupt driven data transfer scheme.

1.5 TIMING DIAGRAM

(AU, Nov/Dec' 19, 13 Marks)

The timing diagram provides information about the various condition (**high** state or **low** state or **high impedance** state) of the signals while a machine cycle is executed. The timing diagrams are supplied by the manufacturer of the microprocessor. The timing diagrams are essential for a system designer. Only from the knowledge of timing diagrams, the matched peripheral devices like memories, ports, etc., can be selected to form a system with microprocessor as CPU.

1.5.1 PROCESSOR CYCLES

The sequence of operations that a processor has to carry out while executing the instruction is called instruction cycle. Each instruction cycle of a processor in turn consists of a number of machine cycles. The machine cycles are the basic operations performed by the processor. To execute an instruction, the processor executes one or more machine cycles in a particular sequence. The machine cycles of a processor are also called processor cycles. The manufacturers of microprocessors define the timings and status of various signals during the processor cycles.

In general, the instruction cycle of an instruction can be divided into two sub cycles: Fetch cycle and Execute cycle. The fetch cycle is executed to fetch the opcode from memory and the execute cycle is executed to decode the instruction and to perform the work specified by the instruction.

1.5.2 MACHINE CYCLES OF 8085

The 8085 microprocessor has seven basic machine cycles. They are as follows:

1. **Opcode fetch cycle (4T or 6T)**
2. **Memory read cycle (3T)**
3. **Memory write cycle (3T)**
4. **IO read cycle (3T)**
5. **IO write cycle (3T)**
6. **Interrupt acknowledge cycle (6T or 12T)**
7. **Bus idle cycle (2T or 3T)**

Each instruction of the 8085 processor consists of one to five machine cycles, i.e., when the 8085 processor executes an instruction, it will execute some of the machine cycles in a specific order. The processor takes a definite time to execute the machine cycles. The time taken by the processor to execute a machine cycle is expressed in T-states.

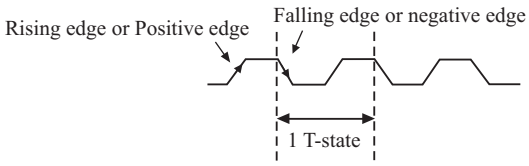


Fig. 1.22: Clock Signal.

One T-state is equal to the time period of the internal clock signal of the processor. The T-state starts at the falling edge of a clock.

Note: Time period, T = 1/f ; where f = Internal clock frequency.

The T-states required by the 8085 processor to execute each machine cycle are mentioned within brackets in the list of machine cycles given above.

1.5.3 OPCODE FETCH MACHINE CYCLE OF 8085

Each instruction of the processor has one-byte opcode. The opcodes are stored in memory. The opcode fetch machine cycle is executed by the processor to fetch the opcode from memory. Hence, every instruction starts with opcode fetch machine cycle.

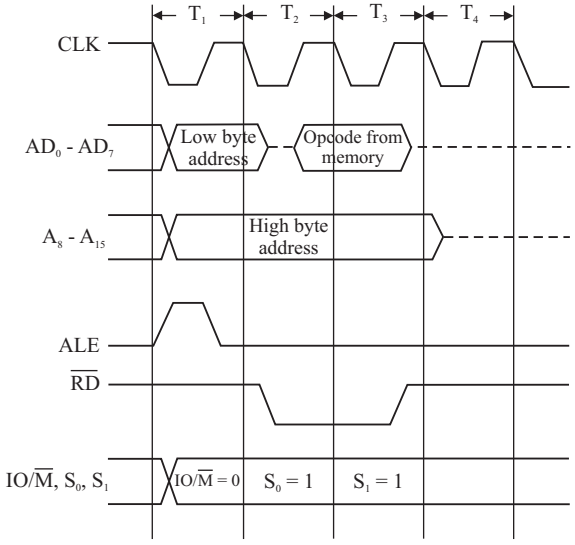
The time taken by the processor to execute the opcode fetch cycle is either 4T or 6T. In this time, the first 3T states are used for fetching the opcode from memory and the remaining T-states are used for internal operations by the processor. The timings of various signals during opcode fetch cycle are shown in Fig. 1.23.

1. **At the falling edge of first T-state (T_1), the microprocessor outputs the low byte address on AD_0-AD_7 lines and high byte address on A_8 to A_{15} lines. ALE is asserted high to enable the external address latch. The other control signals are asserted as follows:**

   $IO/\overline{M}=0$, $S_0 = 1$, $S_1 = 1$. ($IO/\overline{M}$ is asserted low to indicate memory access.)

2. **At the middle of T_1, the ALE is asserted low and this enables the external address latch to take low byte of the address and keep on its output lines.**

3. **In the second T-state (T_2), the memory is requested for read by asserting read line low. When read is asserted low, the memory is enabled for placing the opcode on the data bus. The time allowed for memory to output the opcode is the time during which read remains low.**



($\overline{WR}$ will be **high** ; READY is tied **high** either permanently or temporarily in the system.)

Fig. 1.23: Opcode fetch machine cycle of 8085.

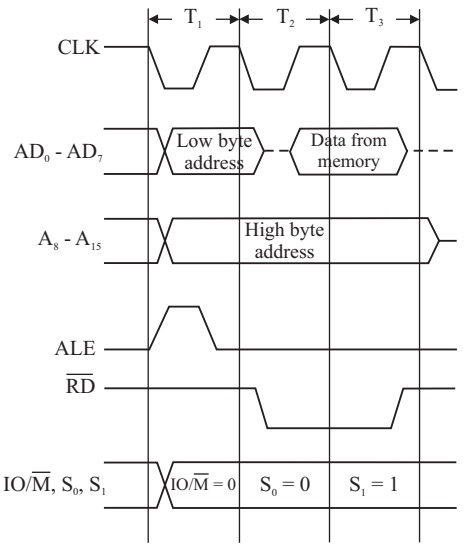
4. **In the third T-state (T_3), the read signal is asserted high. On the rising edge of read signal, the opcode is latched into microprocessor. Other control signals remain in the same state until the next machine cycle.**
5. **The fourth T-state (T_4) is used by the processor for internal operations to decode the instruction and encode into various machine cycles, and also for completing the task specified by 1-byte instruction. During this state (T_4) the address and data bus will be in high impedance state.**

1.5.4 MEMORY READ MACHINE CYCLE OF 8085

The memory read machine cycle is executed by the processor to read a data byte from memory. The processor takes 3T states to execute this cycle. The timings of various signals during memory read cycle are shown in Fig. 1.24.

1. **At the falling edge of T_1, the microprocessor outputs the low byte address on AD_0 - AD_7 lines and high byte address on A_8 to A_{15} lines. ALE is asserted high to enable the external address latch. The other control signals are asserted as follows:**

   $IO/\overline{M}$=0, S_0 = 0, S_1 = 1. ($IO/\overline{M}$ is asserted low to indicate memory access.)
2. **At the middle of T_1, the ALE is asserted low and this enables the external address latch to take low byte of address and keep on its output lines.**
3. **In the second T-state (T_2), the memory is requested for read by asserting read line low. When read is asserted low, the memory is enabled for placing the data on the data bus. The time allowed for memory to output the data is the time during which read remains low.**



($\overline{WR}$ will be **high** ; READY is tied **high** either permanently or temporarily in the system.)

Fig 1.24: Memory read machine cycle of 8085.

4. **At the end of T_3, the read signal is asserted high. On the rising edge of read signal, the data is latched into microprocessor. Other control signals remain in the same state until the next machine cycle.**

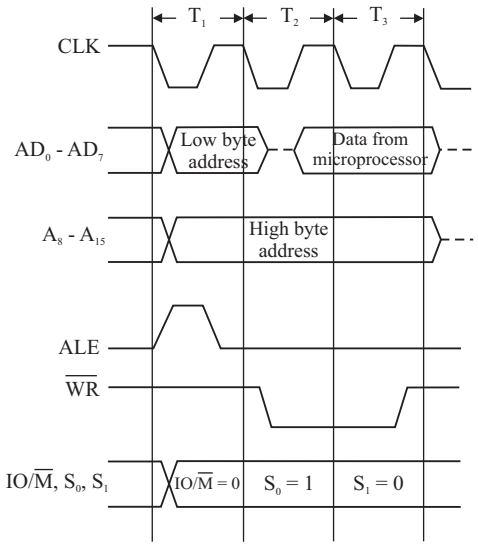
1.5.5 MEMORY WRITE MACHINE CYCLE OF 8085

The memory write machine cycle is executed by the processor to write a data byte in a memory location. The processor takes 3T states to execute this machine cycle. The timings of various signals during memory write cycle are shown in Fig. 1.25.

1. **At the falling edge of T_1, the microprocessor outputs the low byte address on AD_0 - AD_7 lines and high byte address on A_8 to A_{15} lines. ALE is asserted high to enable the external address latch. The other control signals are asserted as follows:**

   $IO/\overline{M}$=0, S_0 = 1, S_1 = 0. ($IO/\overline{M}$ is asserted low to indicate memory access.)

2. **At the middle of T_1, the ALE is asserted low and this enables the external address latch for latching the low byte address into its output lines.**
3. **In the falling edge of T_2, the processor output data on AD_0 to AD_7 lines and then request memory for write operation by asserting the write control signal $\overline{WR}$ to low.**
4. **At the end of T_3, the processor asserts $\overline{WR}$ high. This enables the memory to latch the data into it. The memory should prepare itself to accept the data within the time duration in which write control signal remains low. Other control signals remain in the same state until the next machine cycle.**



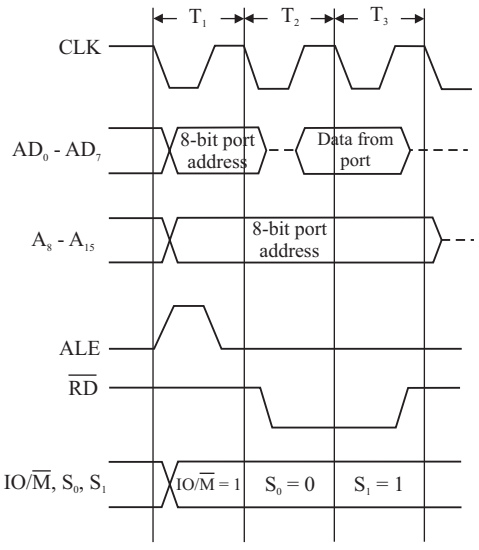
($\overline{RD}$ will be **high** ; READY is tied **high** either permanently or temporarily in the system.)

Fig 1.25: Memory write machine cycle of 8085.

1.5.6 IO READ CYCLE OF 8085

The IO read cycle is executed by the processor to read a data byte from IO port or from the peripheral which is IO-mapped in the system. The processor takes 3T states to execute this machine cycle. The timings of various signals during this machine cycle are shown in Fig. 1.26.

1. **At the falling edge of T_1, the microprocessor output the 8-bit port address on both the low order address lines (AD_0 - AD_7) and high order address lines (A_8 to A_{15}). ALE is asserted high to enable the external address latch. The other control signals are asserted as follows:**
   IO/M = 1, S_0 = 0 and S_1 = 1. (IO/M is asserted high to indicate IO access.)
2. **At the middle of T_1, the ALE is asserted low and this enables the external address latch to take the port address and keep on its output lines.**
3. **In the second T-state (T_2) the IO device is requested for read by asserting read line low. When read is asserted low, the IO port is enabled for placing the data on the data bus. The time allowed for IO port to output the data is the time during which read remains low.**
4. **At the end of T_3, the read signal is asserted high. On the rising edge of read signal the data is latched into microprocessor. Other control signals remains in the same state until the next machine cycle.**



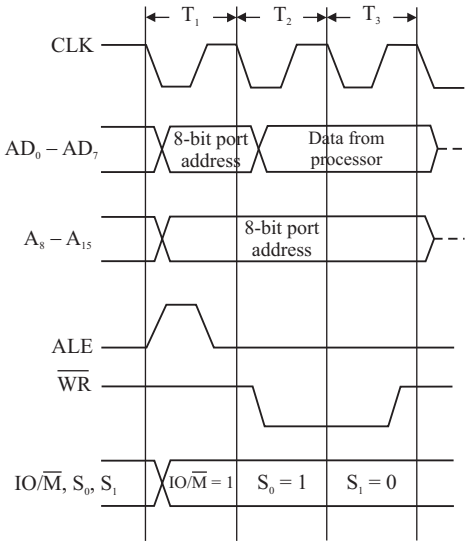
($\overline{WR}$ will be **high** ; READY is tied **high** either permanently or temporarily in the system.)

Fig 1.26: I/O read machine cycle of 8085.

1.5.7 IO WRITE CYCLE OF 8085

The IO write machine cycle is executed by the processor to write a data byte in an IO port or to a peripheral which is IO-mapped in the system. The processor takes 3T states to execute this machine cycle. The timings of the various signals of IO write cycle are shown in Fig. 1.27.

1. **At the falling edge of T_1, the microprocessor outputs the 8-bit port address on low order address line (AD_0 - AD_7) and high order addr$\overline{ess}$ lines (A_8 to A_{15}). ALE is asserted high to enable the external address latch. The other control signals are asserted as follows:**
   IO/$\overline{M}$ = 1, S_0 = 1 and S_1 = 0. (IO/$\overline{M}$ is asserted high to indicate IO access.)
2. **At the middle of T_1, the ALE is asserted low and this enables the external address latch for latching the port address into its output lines.**
3. **In the falling edge of T_2, the processor output data on AD_0 - AD_7 lines and then request IO port for write operation by asserting the write control signal $\overline{WR}$ to low.**
4. **At the end of T_3, the processor asserts $\overline{WR}$ high. This enables the IO port to latch the data into it. The IO port should prepare itself to accept the data within the time duration in which write control signal remains low. Other control signals remains in the same state until the next machine cycle.**



($\overline{RD}$ will be **high** ; READY is tied **high** either permanently or temporarily in the system.)

Fig 1.27: I/O write machine cycle of 8085.

1.5.8 INTERRUPT ACKNOWLEDGE MACHINE CYCLE OF 8085

The interrupt acknowledge machine cycle is executed by the processor to service an interrupt when an interrupt request is made through INTR pin of the processor.

The 8085 processor checks for an interrupt at the second T-state of the last machine cycle of every instruction. If there is a valid interrupt request and if INTR is enabled then the processor completes the current instruction execution and then executes an interrupt acknowledge machine cycle. The interrupt acknowledge machine cycle is executed to get either a **RST n** instruction from the interrupting device or to get a CALL instruction with CALL address from the interrupting device. It also stores the content of program counter (return address) in stack.

1.5.9 INTERRUPT ACKNOWLEDGE CYCLE OF 8085 WITH RST N INSTRUCTION

The timings of various signals during interrupt acknowledge cycle of 8085 when **RST n** instruction is supplied by the interrupting device are shown in Fig. 1.28.

1. **In the first T-state of interrupt acknowledge cycle, the address is placed on the AD_0 - AD_7 and A_8 - A_{15} lines and ALE is asserted high. But the address is not used to read from memory. The other control signals are asserted as follows:**

$IO/\overline{M}=1$, $S_0 = 1$ and $S_1 = 1$.

In the middle of T_1, ALE is asserted low. The INTR signal can remain high or it can go low once the interrupt is accepted.

2. **In the second T-state (T_2), $\overline{INTA}$ is asserted low, and this enables the interrupting device to place the opcode of RST n instruction on the data bus.**

3. **At the end of T_3, the $\overline{INTA}$ is asserted high and the RST n opcode is latched into the processor. The time allowed for the external hardware to place the RST n opcode is the time during which $\overline{INTA}$ remains low.**

4. **The next three T states T_4, T_5 and T_6 are used for internal operations. The internal operations performed are decoding the instruction and encoding into various machine cycles and generation of vector address for the RST n interrupt.**

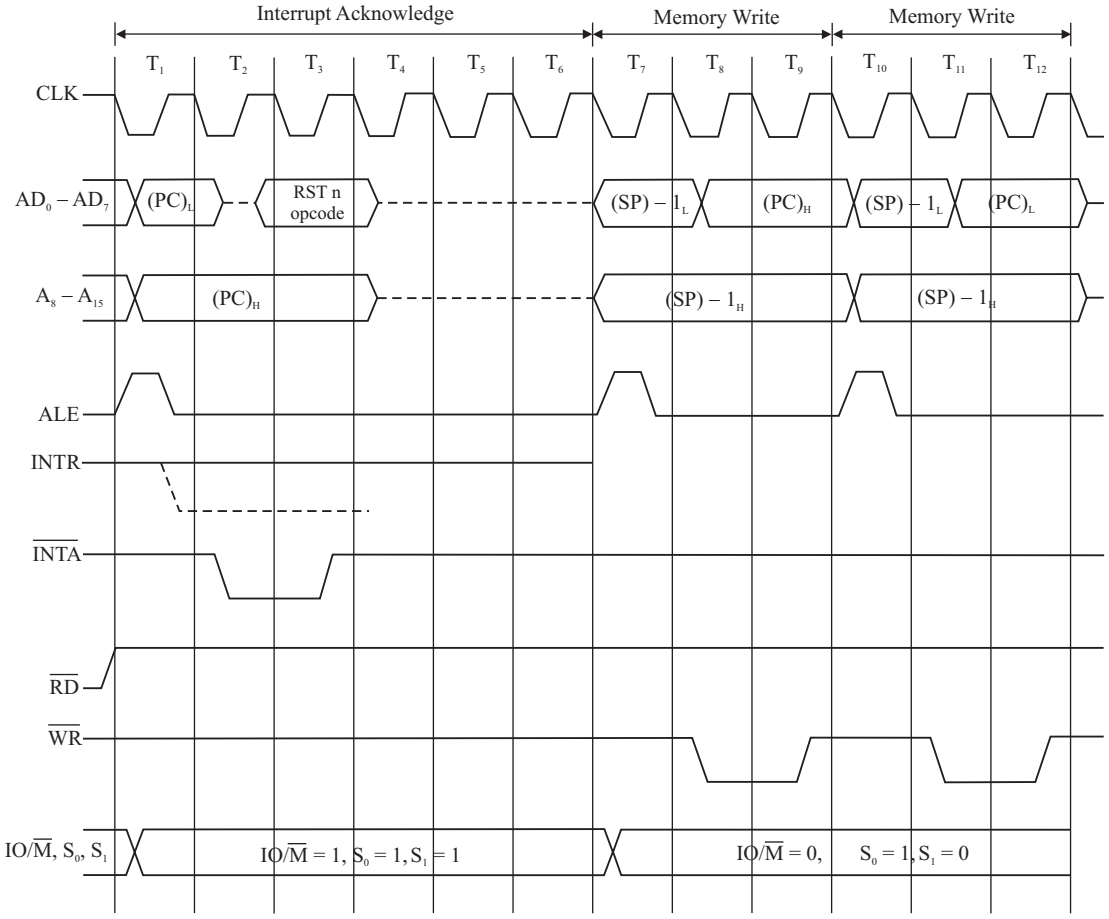


Fig. 1.28: Interrupt acknowledge cycle with **RST n** opcode.

5. **The T states T_7, T_8 and T_9 are used to store the high byte of the Program Counter (PC) in stack (using the content of Stack Pointer (SP) as address).**

In T_7, the content of SP is decremented by one and placed on AD_0-AD_7 and A_8-A_{15} lines. ALE is asserted high and then low, to latch the low byte of address into external latch. The status signals are asserted as IO/$\overline{M}$=0, S_0 = 1 and S_1 = 0.

In T_8, the high byte of PC is placed on AD_0 - AD_7 lines and $\overline{WR}$ is asserted low to enable the stack memory for write operation. At the end of T_9, $\overline{WR}$ is asserted high.

6. **The T states T_{10}, T_{11} and T_{12} are used to store the low byte of the program counter into stack.**

   In T_{10}, the content of SP is again decremented by one and placed on AD_0-AD_7 and A_8 - A_{15} lines. ALE is asserted high and then low, to latch the low byte of address into external latch. The status signals are asserted as IO/$\overline{M}$=0, S_0 = 1 and S_1 = 0.

   In T_{11}, the low byte of PC is placed on AD_0 - AD_7 lines and $\overline{WR}$ is asserted low to enable the stack memory for write operation. At the end of T_{12} $\overline{WR}$ is asserted high.

After the interrupt acknowledge machine cycle, the PC will have the vector address of **RST n** instruction and so the processor starts servicing the interrupt by executing the interrupt service subroutine stored at this address.

1.5.10 INTERRUPT ACKNOWLEDGE CYCLE OF 8085 WITH CALL INSTRUCTION

This cycle is executed by the machine to service an interrupt, when an interrupt request is made through 8259 (Interrupt Controller) to the INTR pin of 8085. The INTEL 8259 can accept 8 interrupt request and allow one by one to the INTR pin of the 8085 processor. It also supplies CALL opcode and CALL address, when it receives INTA signal from the processor.

The processor checks for an interrupt at the second T-state of the last machine cycle of every instruction. If there is a valid interrupt request and if INTR is enabled then the processor completes the current instruction execution and then executes an interrupt acknowledge machine cycle.

The timings of various signals during interrupt acknowledge cycle when CALL instruction is supplied by the interrupting device are shown in Fig. 1.29.

1. **At the falling edge of T_1 the address is placed on AD_0 - AD_7 and A_8 - A_{15} lines and ALE is asserted high. But the address is not used to read from memory. The other control signals are asserted as IO/$\overline{M}$=1, S_0 = 1 and S_1 = 1.**

   In the middle of T_1, ALE is asserted low. The INTR signal can remain high or it can go low once the interrupt is accepted by executing acknowledge cycle.

2. **In T_2, $\overline{INTA}$ is asserted low and this enables the interrupt controller 8259 to place a CALL opcode on the data bus.**

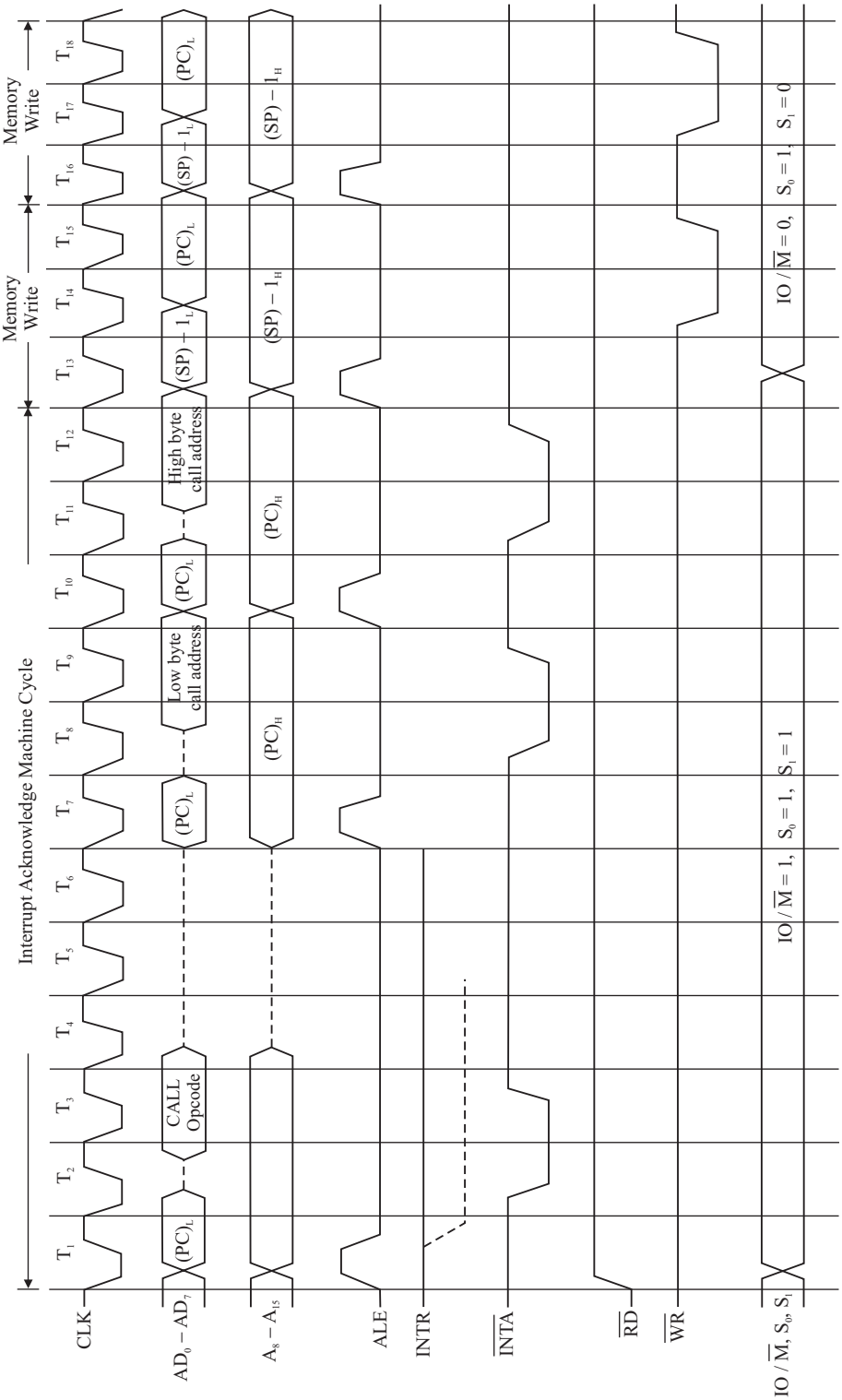


Fig. 1.29: Interrupt acknowledge cycle with CALL opcode.

3. **At the end of T_3, the INTA is asserted high and the CALL opcode is latched into the processor.**

4. **The T states T_4, T_5 and T_6 are used for internal operations. The internal operations performed are decoding the opcode and encoding into various machine cycles.**

5. **The T states T_7, T_8 and T_9 are used to fetch the low byte of call address from 8259. In T_7, the content of Program Counter (PC) is placed on address bus but not used for memory operation. In T_8 the $\overline{INTA}$ is asserted low and this enables the interrupt controller 8259 to place the low byte of call address on data bus. At the end of T_9 the $\overline{INTA}$ is asserted high and the low byte call address on the data bus is latched into the processor.**

6. **The T states T_{10}, T_{11} and T_{12} are used to fetch the high byte of call address from 8259. In T_{10} the content of PC is placed on address bus, but not used for memory operation. In T_{11} the INTA is asserted low and 8259 is enabled for placing the high byte of call address on data bus. At the end of T_{12}, the $\overline{INTA}$ is asserted high and the high byte call address on the data bus is latched into the processor.**

7. **The T states T_{13}, T_{14} and T_{15} are used to store the high byte of the program counter in stack memory. In T_{13}, the content of Stack Pointer (SP) is decremented by one and placed on address bus. ALE is asserted high and then low, to latch the low byte of address into external latch. The other control signals are asserted as IO/$\overline{M}$=0, S_0 = 1 and S_1 = 0. In T_{14}, the high byte of PC is placed on AD_0 - AD_7 lines and $\overline{WR}$ is asserted low to enable the stack memory for write operation. At the end of T_{15}, $\overline{WR}$ is asserted high.**

8. **The T states T_{16}, T_{17} and T_{18} are used to store the low byte of the program counter in stack memory. In T_{16}, the content of SP is again decremented by one and placed on address bus. ALE is asserted high and then low, to latch the low byte of address into external latch. The other control signals are asserted as IO/$\overline{M}$=0, S_0 = 1 and S_1 = 0.**

   In T_{17} the low byte of PC is placed on AD_0 - AD_7 lines and WR is asserted low to enable the stack memory for write operation. At the end of T_{15} $\overline{WR}$ is asserted high.

After the interrupt acknowledge machine cycle, the PC will have the call address and so the processor starts servicing the interrupt by executing the interrupt service subroutine stored at this address.

1.5.11 BUS IDLE MACHINE CYCLE

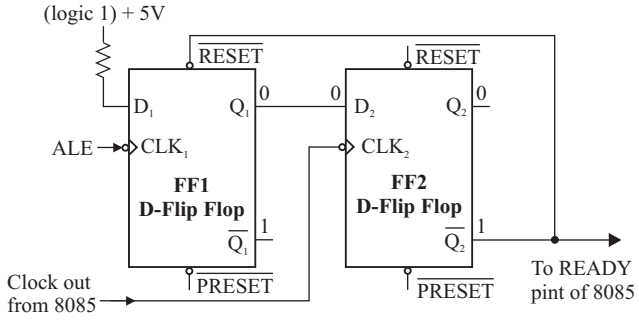
The bus idle machine cycle is executed, when extra time or more time is needed for an internal operation of the processor. During this cycle, the status signals S_0 and S_1 are asserted **low**. The data, address and control pins are driven to **high impedance** state. The READY signal will not be sampled by the processor during this cycle.

1.5.12 MACHINE CYCLE WITH WAIT STATES

Wait states can be introduced in any machine cycle except bus idle cycle between T_2 and T_3. The wait states are introduced in the machine cycle if READY pin is tied **low** at the second T-state of a machine cycle. The processor samples (or check) the READY signal at the second T-state of every machine cycle. If READY is tied **low** at this time, then the processor keeps on introducing wait state until the READY is again tied **high**. This facility is used by the slow memories, IO devices and peripherals to get extra time for read or write operations.

In the system when the peripheral timings are matched with processor timings, then the READY pin is permanently tied **high**. If the system peripherals require more time for read or write cycles, then using additional hardware the READY pin should be tied **low** for the required number of T-states.

The circuit shown in Fig. 1.30 can be used to introduce one wait state in the machine cycles. The working of the circuit shown in Fig. 1.30 can be explained as follows:



(The values shown at the input and output of the flip-flops are initial conditions)

Fig. 1.30: Circuit to introduce one wait state in 8085 machine cycle.

1. **Initially $Q_2 = 0$ and $\overline{Q}_2 = 1$. The input D_1 is permanently tied high. The flip-flops are negative edge sensitive and so they are clocked (recognizes the clock) at the falling edges.**
2. **In the beginning of every machine cycle (except bus idle), ALE is asserted high and then low. At the falling edge of ALE, FF1 is clocked and its output Q_1 changes to 1. Also the input to FF2, D_2 changes to 1.**
3. **Now $D_1 = 1$, $Q_1 = 1$, $D_2 = 1$, $Q_2 = 0$, $\overline{Q}_2 = 1$ and $\overline{RESET} = 1$.**
4. **At the falling edge (beginning) of T_2, FF2 is clocked and so its output Q_2 changes to 1 and changes to 0.**
5. **Now, $D_1 = 1$, $Q_1 = 1$, $D_2 = 1$, $Q_2 = 1$, $\overline{Q}_2 = 0$ and $\overline{RESET} = 0$.**
6. **Since $\overline{Q}_2$ is connected to READY pin of 8085, the READY will be tied low. The $\overline{Q}_2$ is also used to reset FF1 and so when $\overline{Q}_2$ goes to 0 the FF1 is resetted or cleared. Now $Q_1 = 0$ and since $Q_1 = D_2$, the D_2 is also equal to 0.**
7. **Now, $D_1 = 1$, $Q_1 = 0$, $D_2 = 0$, $Q_2 = 1$, $\overline{Q}_2 = 0$ and $\overline{RESET} = 0$.**
8. **At the falling edge of next T-state (i.e., in wait state) again FF2 is clocked and so the output of FF2 will change.**
9. **Now, $D_1 = 1$, $Q_1 = 0$, $D_2 = 0$, $Q_2 = 0$, $\overline{Q}_2 = 1$ and $\overline{RESET} = 1$.**
10. **Since $\overline{Q}_2 = 1$, again READY is tied high. When the processor checks the READY at the falling edge of next cycle (T_3), it will be high and it will continue the machine cycle.**

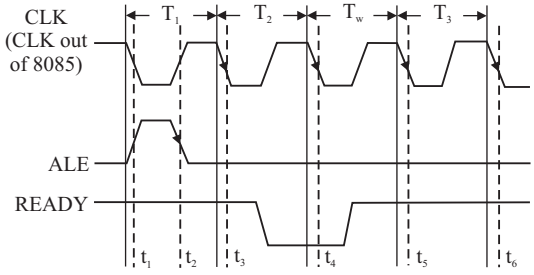
Thus, the hardware shown in Fig. 1.30 introduces one wait state in the machine cycles. A machine cycle with one wait state is shown in Fig. 1.31.

Truth Table of D-flip-flop

Clock	Input	Output
	D	$\overline{Q}$
↓	1	1 0
↓	0	0 1

Preset and reset/clear facility in D-flip-flop

$\overline{PRESET}$	$\overline{RESET}$	Q $\overline{Q}$
0	1	1 0
1	0	0 1
1	1	Clock and D input decide the output
0	0	Should not occur



Note : T_w = *wait state.*

Time	D_1	Q_1	D_2	Q_2	$\overline{Q_2}$	$\overline{RESET}$
t_1	1	0	0	0	1	1
t_2	1	1	1	0	1	1
t_3	1	1	1	1	0	0
	1	0	0	1	0	0
t_4	1	0	0	0	1	1
t_5	1	0	0	0	1	1
t_6	1	0	0	0	1	1

Fig. 1.31: Machine cycle with one wait state.

1.6 INTERRUPTS

Microprocessors allow normal program execution to be interrupted in order to carry out a specific task/work. A processor can be interrupted in the following ways :

(i) by an external signal generated by a peripheral,

(ii) by an internal signal generated by a special instruction in the program,

(iii) by an internal signal generated due to an exceptional condition which occurs while executing an instruction.
(For example, in 8086 processor, 'divide by zero' is an exceptional condition which initiates type-0 interrupt and such an interrupt is also called exception.)

In general, the process of interrupting the normal program execution to carry out a specific task/work is referred to as interrupt.

The interrupt is initiated by a signal generated by an external device or by a signal generated internal to the processor. When a microprocessor receives an interrupt signal, it stops executing the current normal program, saves the status (or content) of various registers (PC in case of 8085) in stack and then executes a subroutine/procedure in order to perform the specific task/work requested by the interrupt. The subroutine/procedure that is executed in response to an interrupt is also called **I**nterrupt

Service **R**outine (ISR). At the end of ISR, the stored status of registers in stack are restored to respective registers and the processor resumes the normal program execution from the point (instruction) where it was interrupted.

The external interrupts are used to implement interrupt driven data transfer scheme. The interrupts generated by special instructions are called software interrupts and they are used to implement system services/calls (or monitor services/calls). The system/monitor services are procedures developed by the system designer for various operations and stored in memory. The user can call these services through software interrupts. The interrupts generated by exceptional conditions are used to implement error conditions in the system.

Interrupt Driven Data Transfer Scheme

Interrupts are useful for efficient data transfer between the processor and the peripheral. When a peripheral is ready for data transfer, it interrupts the processor by sending an appropriate signal. Upon receiving an interrupt signal, the processor suspends the current program execution, saves the status in a stack and executes an ISR to perform the data transfer between the peripheral and the processor. At the end of ISR the processor status is restored from stack and the processor resumes its normal program execution. This type of data transfer scheme is called interrupt driven data transfer scheme.

The data transfer between the processor and peripheral devices can be implemented either by polling technique or by interrupt method. In polling technique, the processor has to periodically poll or check the status/readiness of the device and can perform data transfer only when the device is ready. In polling technique the processor time is wasted, because the processor has to suspend its work and check the status of the device in predefined intervals.

Alternatively, if the device interrupts the processor to initiate a data transfer whenever it is ready then the processor time is effectively utilized because the processor need not suspend its work and check the status of the device in predefined intervals.

For example, consider the data transfer from a keyboard to the processor. Normally a keyboard has to be checked by the processor once in every 10 millisecond for a key press. Therefore, once in every 10 milliseconds the processor has to suspend its work and then check the keyboard for a valid key code. Alternatively, the keyboard can interrupt the processor, whenever a key is pressed and a valid key code is generated. In this way the processor need not waste its time to check the keyboard once in every 10 milliseconds.

1.6.1 CLASSIFICATION OF INTERRUPTS

In general interrupts can be classified in the following three ways:

- **Hardware and software interrupts.**
- **Vectored and non-vectored interrupts.**
- **Maskable and non-maskable interrupts.**

Interrupts initiated by external hardware by sending an appropriate signal to the interrupt pin of the processor is called hardware interrupt. The 8085 processor has five interrupt pins TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR and the interrupts initiated by applying appropriate signal to these pins are called hardware interrupts of 8085.

Software interrupts are program instructions. These instructions are inserted at desired locations in a program. While running a program, if a software interrupt instruction is encountered then the processor initiates an interrupt. The 8085 processor has 8 types of software interrupts. The software interrupt instruction is INT n, where n is the type number in the range 0 to 7.

When an interrupt signal is accepted by the processor, and the program control automatically branches to a specific address (called vector address) then the interrupt is called vectored interrupt. The automatic branching to a vector address is predefined by the manufacturer of the processor. (In these vector addresses the interrupt service subroutines(ISR) are stored.) In non-vectored interrupts the interrupting device should supply the address of the ISR to be executed in response to the interrupt. All the 8085 interrupts excepts INTR are vectored interrupts.

The processors have the facility for accepting or rejecting hardware interrupts. Programming the processor to reject an interrupt is referred to as masking or disabling and programming the processor to accept an interrupt is referred to as unmasking or enabling. In 8085 the hardware interrupts RST 7.5, RST 6.5, and RST 5.5 can be masked/unmasked using SIM instruction. All the hardware interrupts except TRAP are disabled by executing DI instruction and they are enabled by executing EI instruction.

The interrupts whose request can be either accepted or rejected by the processor are called maskable interrupts. The interrupts whose request has to be definitely accepted (i.e., it cannot be rejected) by the processor are called non-maskable interrupts. Whenever a request is made by a non-maskable interrupt, the processor has to definitely accept that request and service that interrupt by suspending its current program and executing an ISR. In 8085 processor all the hardware interrupts except TRAP are maskable. The interrupt initiated through TRAP pin and all software interrupts are non-maskable.

1.6.2 INTERRUPTS OF 8085

The interrupt in 8085 can come from one of the following two sources:

1. **One source is from an external signal applied to TRAP, RST7.5, RST6.5, RST5.5 or INTR pin of the processor. The interrupts initiated by applying appropriate signals to these pins are called hardware interrupts.**
2. **The second source of an interrupt is the execution of the interrupt instruction "RST n" where n can take values from 0 to 7. The interrupts initiated by "RST n" instructions are called software interrupts.**

Software Interrupts Of 8085

Software interrupts are program instructions. When a software interrupt instruction is executed, the processor executes an **I**nterrupt **S**ervice **R**outine(ISR) stored in the vector address of that software

interrupt instruction. The software interrupts of 8085 are RST0, RST1, RST2, RST3, RST4, RST5, RST6 and RST7. The software interrupts of 8085 are vectored interrupts. Software interrupts cannot be masked or be disabled. The Vector addresses of software interrupts are given in Table 1.14.

Table 1.14

Interrupt	Vector address
RST 0	0000_H
RST 1	0008_H
RST 2	0010_H
RST 3	0018_H
RST 4	0020_H
RST 5	0028_H
RST 6	0030_H
RST 7	0038_H

Software interrupt instructions are included at the appropriate (or required) place in the main program. When the processor encounters the software instruction, it pushes the content of PC (**P**rogram **C**ounter) to stack. Then, it loads the vector address in to the PC and starts executing an ISR stored in this address. The last instruction of the ISR will be RET instruction. When the RET instruction is executed, the processor POPs the content of top of stack to PC. Hence, the processor control returns to main program after servicing the interrupt. *[Execution of ISR is referred to as servicing of interrupt.]*

Hardware Interrupts of 8085

(AU, Nov/Dec' 19, 2 Marks)

The hardware interrupts of 8085 are initiated by an external device by placing an appropriate signal at the interrupt pin of the processor. The processor keeps on checking the interrupt pins at the second T-state of the last machine cycle of every instruction. If the processor finds a valid interrupt signal and if the interrupt is unmasked and enabled, then the processor accepts the interrupt. The acceptance of the hardware interrupt is acknowledged by sending an $\overline{\text{INTA}}$ signal to the interrupting device.

When the interrupt is accepted, the processor saves the content of the PC (**P**rogram **C**ounter) in stack and then loads the vector address of the interrupt to the PC. (If the interrupt is non-vectored, then the interrupting device has to supply the address of ISR when it receives $\overline{\text{INTA}}$ signal.) Then the processor starts executing ISR in this address. The last instruction of ISR will be an RET instruction. When the processor executes the RET instruction, it POP the content of top of stack to PC. Thus the processor control returns to the main program after servicing the interrupt.

The hardware interrupts of 8085 are TRAP, RST 7.5, RST6.5, RST5.5 and INTR. TRAP, RST7.5, RST6.5 and RST5.5 are vectored interrupts. In vectored interrupts the address to which the program control is transferred (when the interrupt is accepted) is fixed by the manufacturer. The vector addresses of hardware interrupts are given in Table 1.15. The INTR is a non-vectored interrupt. Hence when a device interrupts through INTR, it has to supply the address of ISR after receiving interrupt acknowledge signal.

Table 1.15

Interrupt	Vector address
RST 7.5	$003C_H$
RST 6.5	0034_H
RST 5.5	$002C_H$
TRAP	0024_H

The type of signal that has to be placed on the interrupt pin of hardware interrupts of 8085 are defined by INTEL. The TRAP interrupt is edge and level sensitive. Hence, to initiate TRAP, the

interrupt signal has to make a **low** to **high** transition and then it has to remain **high** until the interrupt is recognized. The RST 7.5 interrupt is edge sensitive (positive edge). In order to initiate the RST7.5, the interrupt signal has to make a **low** to **high** transition and it need not remain **high** until it is recognized. The RST6.5, RST5.5 and INTR are level sensitive interrupts. Hence, for these interrupts the interrupt signal should remain **high**, until it is recognized.

TRAP is a non-maskable interrupt and RST7.5, RST6.5 and RST5.5 are maskable interrupts, which use the SIM (**S**et **I**nterrupt **M**ask) instruction. Interrupts can be masked by moving an appropriate data (or code) to the accumulator and then executing the SIM instruction. The status of maskable interrupts can be read into the accumulator by executing the RIM instruction (RIM - **R**ead **I**nterrupt **M**ask).

All the hardware interrupts, except TRAP are disabled when the processor is reset and they can also be disabled by executing the DI instruction. (DI - **D**isable **I**nterrupt). When an interrupt is disabled, it will not be accepted by the processor (i.e., INTR, RST5.5, RST6.5 and RST7.5 are disabled by the DI instruction and upon hardware reset). In order to enable (or to allow) the disabled interrupts, the processor has to execute the EI instruction (EI - **E**nable **I**nterrupt).

Priorities of Interrupts of 8085

When all the interrupts are enabled, the priority sequence of hardware interrupts from highest to lowest is TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR. When the 8085 processor accepts an interrupt it will disable all the hardware interrupts except TRAP. Hence in order to allow the higher priority interrupt while executing Interrupt Service Subroutine (ISR) for lower priority interrupt, enable the interrupt system in the beginning of ISR of lower priority interrupt, by executing EI instruction.

For example, if the processor accepts RST 5.5 interrupt, then it will disable RST 7.5, RST 6.5 and INTR interrupts. In order to allow the higher priority interrupt RST 7.5 and RST 6.5 while executing ISR of RST 5.5, the EI instruction should be executed in the beginning of ISR of RST 5.5.

The execution of software interrupt will not disable any hardware interrupt. Therefore while executing ISR of software interrupts, the processor will recognize or allow the hardware interrupts.

1.6.3 ENABLING, DISABLING AND MASKING OF 8085 INTERRUPTS

TRAP

The interrupt TRAP is non-maskable and it cannot be disabled by DI instruction. Also the TRAP is not disabled by system (processor) reset or after recognition of another interrupt. The only signal which can override TRAP is HOLD signal. (i.e., If the processor receives HOLD and TRAP at the same time then HOLD is recognized first and only then is TRAP recognized.)

INTR

The interrupt INTR is disabled by any one of the following operations:

- **Executing DI instruction.**
- **System or processor reset.**
- **After recognition (acceptance) of an interrupt.**

The interrupt INTR can be enabled by executing EI instruction.

RST 7.5, RST 6.5 and RST 5.5

The interrupt RST 7.5, RST 6.5 and RST 5.5 are disabled by any one of the following operations.

- **Executing DI instruction.**
- **System or processor reset.**
- **After recognition (acceptance) of an interrupt.**

These hardware interrupts can be enabled by executing EI instruction.

The 8085 provides additional masking facility for RST 7.5, RST 6.5 and RST 5.5 using SIM instruction. The status of these interrupts can be read by executing RIM instruction.

The masking or unmasking of RST 7.5, RST 6.5 and RST 5.5 interrupts can be performed by moving an 8-bit data to accumulator and then executing SIM instruction. The format of the 8-bit data is shown in Fig. 1.32.

The status of pending interrupts can be read from accumulator after executing RIM instruction. When RIM instruction is executed, an 8-bit data is loaded to the accumulator, which can be interpreted as shown in Fig. 1.33.

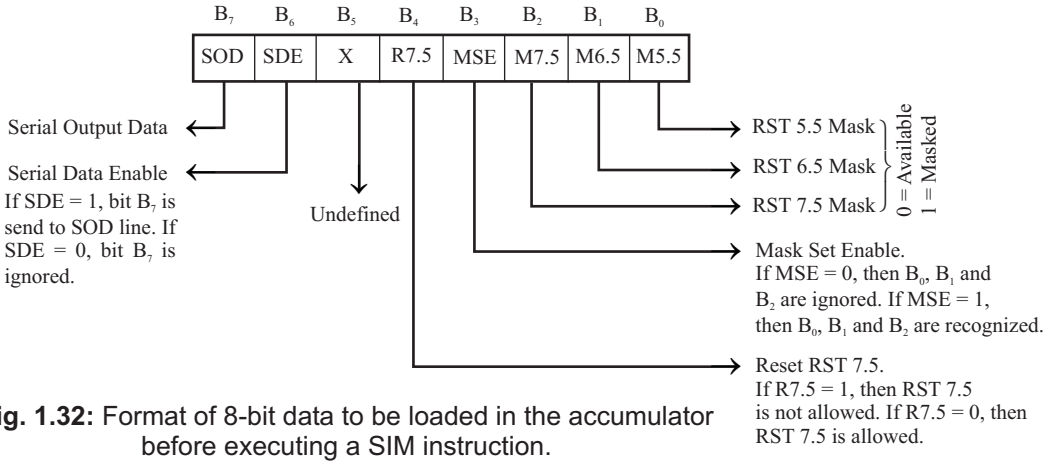


Fig. 1.32: Format of 8-bit data to be loaded in the accumulator before executing a SIM instruction.

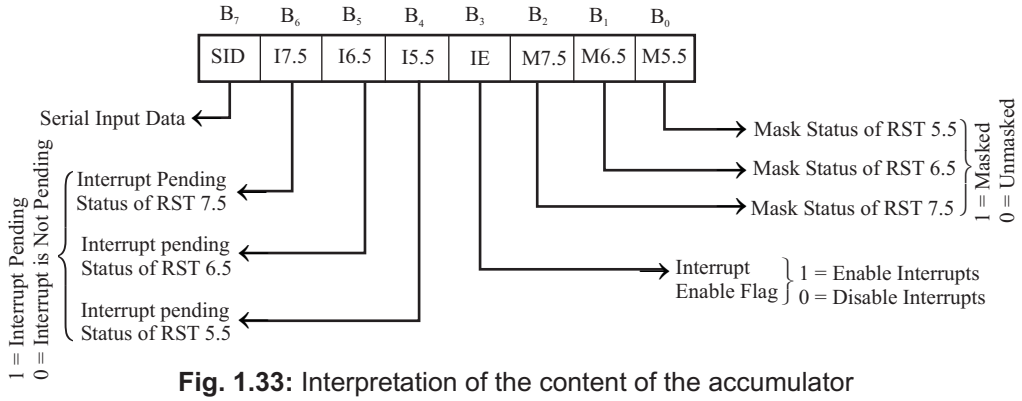


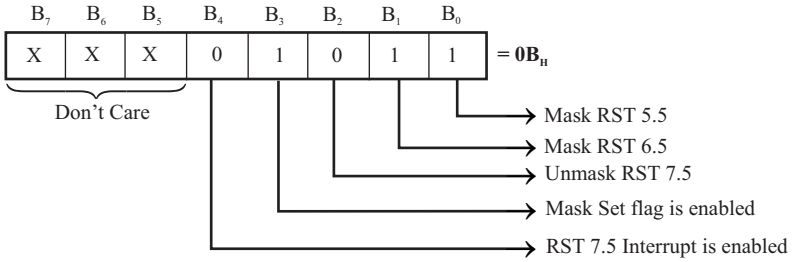
Fig. 1.33: Interpretation of the content of the accumulator after executing a RIM instruction.

EXAMPLE 1

Write a program segment to mask RST 6.5 and RST 5.5 interrupts and enable RST 7.5 interrupt.

Solution

The 8-bit data format to be loaded in the accumulator for enabling RST 7.5 and masking RST 6.5 and RST 5.5 is shown below. The data to be loaded in accumulator is $0B_H$.



Program segment

```
EI          ; Enable all interrupts of 8085
MVI A,0BH   ; Move 0BH to A-register
SIM         ; Mask 6.5 and 5.5, Enable 7.5
```

EXAMPLE 2

Assume that the 8085 microprocessor returns to the main program after servicing RST 6.5. (Remember that while servicing an interrupt all other interrupts are disabled.)

Write a program segment to check whether RST 5.5 interrupt is pending. If it is pending then the program has to enable RST 5.5 without affecting any other interrupts. Otherwise the program has to enable all interrupts and return to main program.

Solution

The status of pending interrupts can be read by executing RIM instruction. This will load an 8-bit data in accumulator. If RST 5.5 is pending then the bit D_4 in accumulator will be **1** and if it is not pending then bit D_4 will be **0**. The following program segment have been written to check whether bit D_4 is **1** or **0.** If it is **1** then the program control jumps to another part of program to enable RST 5.5 and mask other interrupts.

Program segment

```
        RIM       ;Read the status of interrupts.
        MOV C,A   ;Save the status in C-register.
        ANI 10H   ;Check whether RST 5.5 is pending.
        JNZ NEXT  ;If  RST 5.5 is pending, go to NEXT.
        EI        ;If RST 5.5 is not pending, enable
        RET       ;all interrupts and return to main program.
NEXT:   MOV A,C   ;Get the interrupt status in A-register.
        ANI FEH   ;Set D0 = 0, for enabling RST 5.5
        ORI 08H   ;Set D3 =1, for enabling interrupt enable flag.
        SIM       ;Enable RST 5.5
        JMP ISR55 ;Jump to Interrupt Service Routine of RST 5.5
```

1.6.4 INTR AND ITS EXPANSION

The INTR is general interrupt request. An external device can interrupt the processor by placing a **high** signal on INTR pin of 8085. If the processor accepts the interrupt, then it will send an acknowledge signal $\overline{INTA}$ to the interrupting device. On receiving the acknowledge signal, the interrupting device has to place either an **RST n** opcode (or CALL opcode followed by 16-bit address) on the data bus.

On receiving the **RST n** opcode, the 8085 processor generates the vector address of **RST n** instruction. It saves the content of **P**rogram **C**ounter (PC) in stack. Then it loads the vector address in PC and executes an **I**nterrupt **S**ervice **R**outine (ISR) stored at this address. (when it receives CALL opcode it executes an interrupt service routine stored at CALL address.)

The INTR interrupt can be expanded to accept 8-interrupt inputs using 8-to-3 priority encoder as shown in Fig. 1.34.

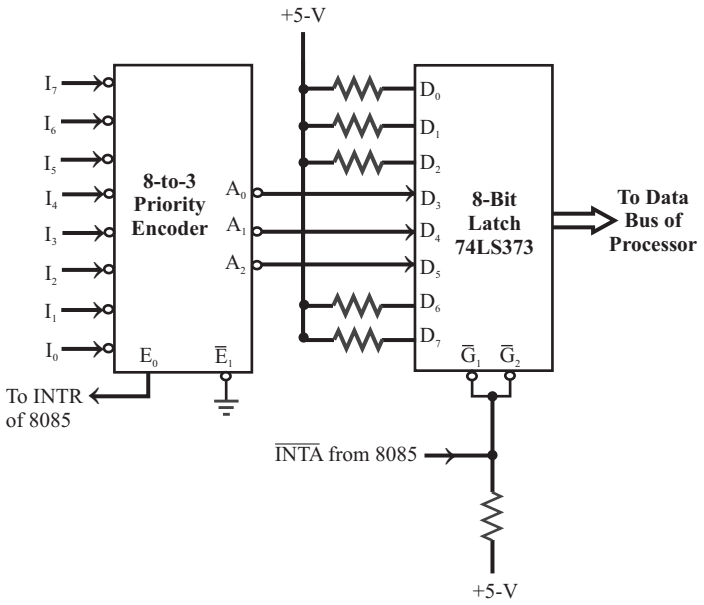


Fig. 1.34: Expanding an INTR of the 8085 using an 8-to-3 priority encoder.

The priority encoder has 8 inputs I_0 to I_7 and three outputs A_0 to A_2. It also has an output control signal, E_0. If the priority encoder receives a logic **low** at one of the inputs, for example I_n, then it asserts E_0 **high** and outputs the binary value of n on the output lines A_0, A_1 and A_2 lines (i.e., if input I_0 is **low** then output is 000; if input I_1 is **low** then output is 001 and so on). In this scheme I_7 has the highest priority and I_0 has the lowest priority.

Eight external devices can interrupt the processor through I_0 to I_7 lines, by placing a logic **low** on these pins. On receiving a valid interrupt signal the priority encoder allows the highest priority interrupt by asserting E_0 **high** and sending the corresponding binary value on A_0, A_1 and A_2 lines. The E_0 is connected to INTR of 8085 and A_0, A_1 and A_2 are connected to the inputs D_3, D_4 and D_5 of an 8-bit latch. All other inputs of the latch are tied to +5 V (logic 1) permanently.

The opcodes and vector addresses of RST n instructions are shown in Table 1.16. If we carefully look at the opcode of RST instruction, the binary bits D_3, D_4, D_5 constitutes the binary value of n in RST n instruction and all other bits are 1's. The priority encoder helps in placing the RST opcodes at the input of latch (74LS373). *[The priority encoder places the **RST n** opcode for the interrupt I_n.]*

When the processor accepts the interrupt, it sends $\overline{\text{INTA}}$ signal to the interrupting device. This signal is used to enable the latch. When the latch is enabled, the RST opcode available at the input is latched into output lines. The output of latch is connected to data bus of the processor. Hence, the opcode will be placed on the data bus.

Table 1.16: Opcodes of RST Instructions

RST instruction	Opcode in binary								Opcode in hexa	Vector address
	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0		
RST 0	1	1	0	0	0	1	1	1	C7	0000_H
RST 1	1	1	0	0	1	1	1	1	CF	0008_H
RST 2	1	1	0	1	0	1	1	1	D7	0010_H
RST 3	1	1	0	1	1	1	1	1	DF	0018_H
RST 4	1	1	1	0	0	1	1	1	E7	0020_H
RST 5	1	1	1	0	1	1	1	1	EF	0028_H
RST 6	1	1	1	1	0	1	1	1	F7	0030_H
RST 7	1	1	1	1	1	1	1	1	FF	0038_H

This opcode is read by the processor and then it generates the vector address of the RST instruction internally. The processor saves the current value of **P**rogram **C**ounter (PC) in stack and loads the vector address in PC. Now the processor starts servicing the interrupt.

1.7 SHORT-ANSWER QUESTIONS

Q1.1 What is a microprocessor ?

A microprocessor is a program controlled semiconductor device (IC), which fetches, decodes and executes instructions.

Q1.2 What are the basic functional blocks of a microprocessor ?

The basic functional blocks of a microprocessor are ALU, an array of registers and control unit.

Q1.3 What is a bus ?

Bus is a group of conducting lines that carries data, addresses and control signals.

Q1.4 Define bit, byte and word.

A digit of the binary number or code is called bit. The bit is also the fundamental storage unit of computer memory.

The 8-bit (8-digit) binary number or code is called byte and 16-bit binary number or code is called word. (Some microprocessor manufacturers refer to the basic data size operated by the processor as word.)

Q1.5 State the relation between the number of address pins and physical memory space?

The size of the binary number used to address the memory decides the physical memory space. If a microprocessor has n-address pins then it can directly address 2^n memory locations. (The memory locations that are directly addressed by the processor are called physical memory space.)

Q1.6 Why is data bus is bidirectional?

The microprocessor has to fetch (read) the data from memory or input device for processing and after processing it has to store (write) the data in memory or output device. Hence, the data bus is bidirectional.

Q1.7 Why is address bus unidirectional?

The address is an identification number used by the microprocessor to identify or access a memory location or IO device. It is an output signal from the processor. Hence, the address bus is unidirectional.

Q1.8 State the difference between CPU and ALU.

The ALU is the unit that performs the arithmetic or logical operations. The CPU is the unit that includes ALU and control unit. Apart from processing the data, the CPU controls the entire system functioning. Usually, a microprocessor will be the CPU of a system and it is called the brain of the computer.

Q1.9 What is a tristate logic? Why it is needed in microprocessor system?

In a tristate logic, three logic levels are used **high, low** and **high impedance** state. The **high** and **low** are normal logic levels and **high impedance** state is electrical open circuit condition.

In a microprocessor system, all the peripheral/slave devices are connected to a common bus. But communication (data transfer) takes place between the master (microprocessor) and one slave (peripheral) at any time instant. During this time instant, all other devices should be isolated from the bus. Therefore, normally all the slaves (peripherals) will remain in **high impedance** state (i.e., in electrical isolation). The master will select a slave by sending address and chip select signal. When the slave is selected, it comes to normal logic and it can communicate with the master.

Q1.10 What is HMOS and HCMOS.

The HMOS is High density n-type Metal Oxide Silicon field effect transistors. The third generation microprocessors are fabricated using HMOS transistors.

The HCMOS is High density n-type Complementary Metal Oxide Silicon field effect transistors. It is the low power version of HMOS and the fourth generation microprocessors are fabricated using HCMOS transistors.

Q1.11 What are the drawbacks of first generation microprocessors.

The first generation processors are fabricated using PMOS technology and it has the drawbacks like slow speed, provides low output currents and was not compatible with TTL logic levels.

Q1.12 What is a microcomputer? Explain the difference between a microprocessor and a microcomputer.

A system designed using a microprocessor as its CPU is called microcomputer. The term microcomputer refers to the whole system, whereas the microprocessor is the CPU of the system.

Q1.13 What is the function of microprocessor in a system?

The microprocessor is the master in the system, which controls all the activity of the system. It issues address and control signals and fetches the instruction and data from memory. Then it executes the instruction to take appropriate action.

Q1.14 List the components of microprocessor-based (single board microcomputer) system.

The microprocessor-based system consist of microprocessor as CPU, semiconductor memories like EPROM and RAM, input device, output device and interfacing devices.

Q1.15 Why interfacing is needed for IO devices? *(AU, Nov/Dec' 19, 2 Marks)*

Generally IO devices are slow devices. Therefore, the speed of IO devices does not match with the speed of microprocessor. And so an interface is provided between system bus and IO devices.

Q1.16 What is the difference between CPU bus and system bus?

The CPU bus has multiplexed lines but the system bus has separate lines for each signal. (The multiplexed CPU lines are demultiplexed by the CPU interface circuit to form system bus.)

Q1.17 What is multiplexing and what is its advantage?

Multiplexing is transferring different information at different well-defined times through same lines. A group of such lines is called multiplexed bus. The advantage of multiplexing is that fewer pins are required for microprocessors to communicate with the outside world.

Q1.18 How the address and data lines are demultiplexed in 8085?

The low order address and data lines of 8085 are demultiplexed using an external 8-bit D-Latch (74LS373) and the ALE signal of 8085, as shown in Fig. Q1.18.

At the beginning of every machine cycle, ALE is asserted **high** and then **low**. Also, the low byte of address is given out through AD_0 - AD_7 lines. Since, the ALE is connected to enable of latch, whenever ALE is asserted **high** and then **low,** the addresses are latched into the output lines of the latch then the lines AD_0 - AD_7 are free for data transfer.

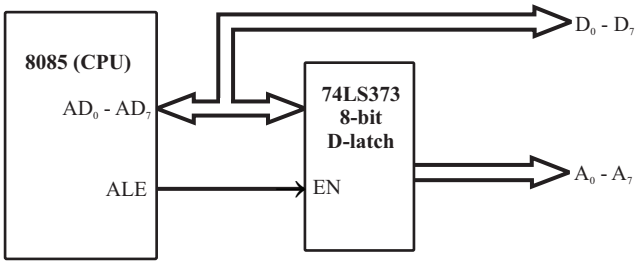


Fig. Q1.18: Demultiplexing of address and data lines in an 8085 processor.

Q1.19 On what basis the computers are classified into micro, mini and large/mainframe?

The classification of computers into micro, mini and mainframes are based on the following factors:

1. **Speed of execution**
2. **Size and type of data**
3. **Memory capacity**
4. **IO devices and peripheral support devices**
5. **Application programs it can run.**

Q1.20 What is the difference between micro and minicomputers?

The microcomputers are systems built using a single microprocessor and has single motherboard as CPU. The minicomputers will have multiple microprocessors connected in a particular configuration. The minicomputer can handle large data words, address more memory space, supports a variety of IO devices and can be used for multiuser applications. In today's technology, the features of microcomputers have exceeded the capability of minicomputers.

Q1.21 What is mainframe?

The largest and most powerful computers are called mainframes.

Q1.22 What is a supercomputer?

The computer built using very high speed devices (or devices with very low switching speeds) and can execute instructions at very high speeds are called supercomputers. The speed of supercomputers are measured in MIPS (**M**illions of **I**nstructions **P**er **S**econd) or Megaflops (Millions of floating point operations per second). A typical supercomputer can execute 3000 MIPS. The speed of Cray X-MP2 supercomputer is 500 Megaflops.

Q1.23 List the applications of microcomputer.

1. **Personal computing**
2. **Calculators**
3. **Small business system.**
4. **Control applications**
5. **Instrumentation systems**

Q1.24 What are the advantages of microprocessor-based system?

The advantages of microprocessor-based system are the following:

1. **Computational or Processing speed is high.**
2. **Intelligence has been brought to systems.**
3. **Automation of industrial processes and office administration.**
4. **Both operation and maintenance are easier.**

Q1.25 What are the disadvantages of microprocessor-based system?

The following are the disadvantages of microprocessor-based system:

1. **It has limitations on the size of data.**
2. **The applications are limited by the physical address space.**
3. **The analog signals cannot be processed directly and digitizing the analog signals introduces errors.**
4. **Most of the microprocessors do not support floating point operations.**

Q1.26 What do you mean by 16 and 8-bit processors? Mention a few 8-bit and 16-bit processors.

The processors are classified into 8-bit or 16-bit depending on the basic data size handled by the ALU of the processor.

8-bit microprocessors : 8085, Z80, Motorola 6800.
16-bit microprocessors : 8086, Z8000, MC68000.

Q1.27 What is the fabrication technology used for 8085?

The 8085A is fabricated used NMOS technology and 8085AH is fabricated using HMOS technology.

Q1.28 What is the physical memory space in 8085?

The 8085 uses 16-bit address to access memory locations. Hence, it can directly address 64 k memory locations (2^{16} = 65,536 = 64 k). Since 8085 has 8 data lines, it can read or write 8-data bits from a memory address. Therefore, the physical memory space is 64 k × 1byte = 64 kilobytes (64 kB).

Q1.29 What is ALE?

The ALE (**A**ddress **L**atch **E**nable) is a signal used to demultiplex the address and data lines using an external latch. It is used as enable signal for the external latch.

Q1.30 Explain the function of IO/$\overline{M}$ in 8085.

The IO/$\overline{M}$ is used to differentiate memory access and IO access. For IN and OUT instruction it is asserted **high**. For memory reference instructions it is asserted **low**.

Q1.31 How the READY signal is used in microprocessor system?

The READY is an input signal that can be used by slow peripherals to get extra time in order to communicate with 8085. The 8085 will work only when READY is tied to logic **high**. Whenever READY is tied to logic **low**, the 8085 will enter a wait state. When the system has slow peripheral devices, additional hardware is provided in the system to make the READY input **low** during the required extra time while executing a machine cycle, so that the processor will remain in wait state during this extra time.

Q1.32 What is HOLD and HLDA? How is it used?

The HOLD and HLDA signals are used for the **D**irect **M**emory **A**ccess (DMA) type of data transfer. These type of data transfers are achieved by employing a DMA controller in the system. When DMA is required, the DMA controller will place a **high** signal on the HOLD pin of 8085. When HOLD input is asserted **high,** the processor will enter a wait state and drive all its tristate pins to **high impedance** state and send an acknowledge signal to DMA controller through HLDA pin. Upon receiving the acknowledge signal, the DMA controller will take control of the bus and perform DMA transfer and at the end it asserts HOLD signal **low**. When HOLD is asserted **low,** the processor will resume its execution.

Q1.33 How clock signals are generated in 8085 and what is the frequency of the internal clock?

The 8085 has the clock generation circuit on the chip but an external quartz crystal or LC circuit or RC circuit should be connected at the pins X_1 and X_2 in order to generate a clock signal. The 8085 clock generation circuit, generate a clock whose frequency is double as compared to that of internal clock. The generated clock is divided by two and then used as internal clock. The maximum internal clock frequency of 8085A is 3.03 MHz.

Q1.34 What happens to the 8085 processor when it is resetted?

When $\overline{\text{RESET IN}}$ pin is asserted **low**, the program counter, instruction register, interrupt mask bits and all internal registers are cleared or resetted. Also the RESET OUT signal is asserted **high** to clear or reset all the peripheral devices in the system. After a reset, the content of program counter will be 0000_H and so the processor will start executing the program stored at 0000_H.

Q1.35 What are the operations performed by ALU of 8085?

The operations performed by ALU of 8085 are addition, subtraction, logical AND, OR, Exclusive-OR, compare, complement, increment, decrement and left/right shift.

Q1.36 Mention the names of various registers in 8085 along with its size.

Register		Size (bits)	Register		Size (bits)
Accumulator (A)	-	8	Stack pointer	-	16
Temporary register	-	8	Program counter	-	16
Instruction register	-	8			
General purpose register (B, C, D, E, H and L)	-	8			

Q1.37 What is a flag?

Flag is a flip- flop used to store the information about the status of the processor and the status of the instruction executed most recently.

Q1.38 List the flags of 8085.

There are five flags in 8085. They are sign flag, zero flag, auxiliary carry flag, parity flag and carry flag.

Q1.39 Show the bit positions of various flags in 8085 flag register.

The bit positions of various flags in the flag register of 8085 is shown in Fig. Q1.39.

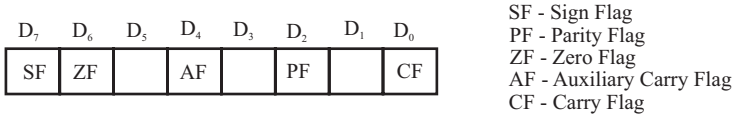


Fig. Q1.39: Bit positions of various flags in the flag register of 8085.

Q1.40 What are the hardware interrupts of 8085?

The hardware interrupts in 8085 are TRAP, RST 7.5, RST 6.5 and RST 5.5.

Q1.41 Which interrupt has highest priority in 8085? What is the priority of other interrupts?

The TRAP has the highest priority, followed by RST 7.5, RST 6.5, RST 5.5 and INTR.

Q1.42 Define stack.

Stack is a sequence of RAM memory locations defined by the programmer.

Q1.43 What is program counter? How is it useful in program execution?

The program counter keeps a track of program execution. To execute a program, the starting address of the program is loaded in program counter. The PC sends out an address to fetch a byte of instruction from memory and increment its content automatically.

Q1.44 How is the microprocessor synchronized with peripherals ?

The timing and control unit synchronizes all the microprocessor operations with clock and generates control signals necessary for communication between the microprocessor and peripherals.

Q1.45 What is memory ?

A memory is a storage device in a microprocessor-based system and its primary function is to store programs and data.

Q1.46 What is physical memory space?

The memory locations that are directly addressed by the microprocessor is called physical memory space.

Q1.47 What is memory word size?

The size of data that can be stored in memory location is called memory word size.

Q1.48 What is meant by memory mapping?

Memory mapping is the process of interfacing memories to microprocessor and allocating addresses to each memory location.

Q1.49 What is memory access time?

Memory access time is the time taken by the processor to read or write a memory location. Read operation is the time between a valid address on the bus and the end of read control signal. Write operation is the time between a valid address on the bus and the end of write control signal.

Q1.50 What is bus contention?

If two devices drive the data bus simultaneously then it is called bus contention. It may lead to following undesirable events:

1. **Damaging one or both the IC chip.**
2. **The high current may cause a voltage spike in the supply system leading to data loss.**

Q1.51 Why is EPROM mapped at the beginning of memory space in 8085 ?

When EPROM is mapped at the beginning of memory space, then 0000_H address will be allotted to EPROM. The monitor program can be stored from 0000_H address. Whenever the processor is reset, the program counter will be cleared (i.e it will have 0000_H address) and the monitor program will be executed automatically.

Q1.52 What is chip select and how it is generated?

Chip select is the control signal that has to be asserted TRUE to bring an IC from **high impedance** state to normal state. Generally the chip select signals are generated in a system by decoding the unused address lines with the help of decoders.

Q1.53 What are the typical control signals involved in EPROM interfacing ?

The control signals needed for EPROM are chip select and output enable.

Q1.54 What are the typical control signals involved in RAM interfacing ?

The control signals needed for RAM interfacing are chip enable, output enable and write enable.

Q1. 55 What is programmed IO ?

If the data transfer between an IO device and the processor is accomplished through an IO port and controlled by a program then the IO device is called programmed IO.

Q1.56 What is interrupt IO?

If the IO device initiates the data transfer through interrupt, then the IO is called interrupt driven IO.

Q1.57 What is DMA?

The direct data transfer between the IO device and the memory is called DMA.

Q1.58 What is the need for Port?

IO devices are generally slow devices and their timing characteristics do not match with processor timings. Hence, the IO devices are connected to a system bus through the ports.

Q1.59 What is a port?

A port is a buffered I/C which is used to hold the data transmitted from the microprocessor to IO device or vice versa.

Q1.60 Give some examples of port devices used in a 8085 microprocessor-based system.

The various INTEL IO port devices used in 8085 microprocessor-based system are 8212, 8155, 8156, 8255, 8355 and 8755.

Q1.61 What are the different methods of interfacing IO devices to 8085-based system.

There are two methods of interfacing IO devices to 8085 system. They are memory mapping of IO device and standard IO mapping.

Q1.62 Compare the memory-mapped IO with the standard IO-mapped IO.

Memory-mapped IO	Standard IO-mapped IO
1. 16-bit address is allotted to an IO device.	1. 8-bit address is allotted to an IO device.
2. The devices are accessed by memory read or memory write cycle.	2. The devices are accessed by IO read or IO write cycle.
3. All instructions related to memory can be used for data.	3. Only IN and OUT instructions can be used for data transfer.
4. A large number of IO ports can be interfaced.	4. Only 256 ports can be interfaced.

Q1.63 What is the drawback in memory-mapped IO?

When IO devices are memory-mapped, some of the addresses are allotted to IO devices. So the full address space cannot be used for addressing memory (i.e., physical memory address space will be reduced). Hence, memory mapping is useful only for small systems, where the memory requirement is less.

Q1.64 What is Software and Hardware?

The software is a set of instructions or commands needed for performing a specific task by a programmable device or a computing machine.

The hardware refers to the components or devices used to form computing machine in which the software can be run and tested. Without software the hardware is an idle machine.

Q1.65 What is machine language?

The language that can be understood by a programmable machine is called machine language. The machine language program are developed using 1s and 0 s.

Q1.66 What is assembly language?

The language in which the mnemonics (short-hand form of instructions) are used to write a program is called assembly language. The mnemonics are given by the manufacturers of microprocessor.

Q1.67 Draw a simple circuit to decode the three control signals $\overline{RD}$,$\overline{WR}$ and IO/$\overline{M}$ and to produce separate read/write control signals for memory and IO devices.

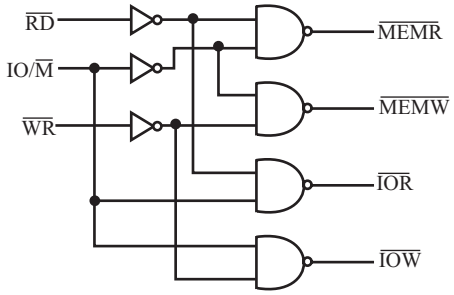


Fig. Q1.67: Circuit to generate separate read and write signals for memory and IO devices in an 8085-based system.

Q1.68 What are machine language and assembly language programs?

The software developed using 1s and 0 s are called machine language programs. The software developed using mnemonics are called assembly language programs.

Q1.69 What is the drawback in machine language and assembly language programs?

The machine language and assembly language programs are machine dependent. The programs developed using these languages for a particular machine cannot be directly run on another machine. (But after conversion using suitable conversion software it can be run on another machine.)

Q1.70 Define mnemonics.

The short-hand form of describing the instructions are called mnemonics. The mnemonics are given by the manufacturers of microprocessors and programmable devices.

Q1.71 What is processor cycle (machine cycle)?

The processor cycle or machine cycle is the basic external operation performed by the processor. To execute an instruction, the processor will run one or more machine cycles in a particular order.

Q1.72 What is instruction cycle?

The sequence of operations that a processor has to carry out while executing an instruction is called instruction cycle. Each instruction cycle of a processor in turn consists of a number of machine cycles.

Q1.73 What is fetch and execute cycle?

In general, the instruction cycle of an instruction can be divided into fetch and execute cycles. The fetch cycle is executed to fetch the opcode from memory. The execute cycle is executed to decode the instruction and to perform the work instructed by the instruction.

Q1.74 List the various machine cycles of 8085.

The various machine cycles of 8085 are as follows:

(i) Opcode fetch cycle

(ii) Memory read cycle

(iii) Memory write cycle

(iv) IO read cycle

(v) IO write cycle

(vi) Interrupt acknowledge cycle

(vii) Bus idle cycle.

Q1.75 What is the need for timing diagram?

The timing diagram provides information regarding the status of various signals, when a machine cycle is executed. The knowledge of timing diagram is essential for system designer to select matched peripheral devices like memories, latches, ports, etc., to form a microprocessor system.

Q1.76 What is T-state?

The T-state is the time period of the internal clock signal of the processor. The time taken by the processor to execute a machine cycle is expressed in T-state.

Q1.77 How many machine cycles constitute one instruction cycle in 8085?

Each instruction of the 8085 processor consist of one to five machine cycles.

Q1.78 Define opcode and operand.

Opcode (**Op**eration **Code**) is the part of an instruction/directive that identifies a specific operation.

Operand is a part of an instruction/directive that represents a value on which the instruction acts.

Q1.79 What is opcode fetch cycle?

The opcode fetch cycle is a machine cycle executed to fetch the opcode of an instruction stored in memory. The first machine cycle of every instruction is opcode fetch machine cycle.

Q1.80 What operation is performed during first T-state of every machine cycle in 8085?

In 8085, during the first T-state of every machine cycle the low byte address is latched into an external latch using ALE signal.

Q1.81 Why status signals are provided in microprocessor?

The status signals can be used by the system designer to track the internal operations of the processor. Also, it can be used for memory expansion (by providing separate memory banks for program and data, and selecting the banks using status signals).

Q1.82 How the 8085 processor differentiates memory access (read/write) and IO access (read/write)?

The memory access and IO access is differentiated using IO/$\overline{M}$ signal. The 8085 processor asserts IO/$\overline{M}$ **low** for memory read/write operation and IO/$\overline{M}$ is asserted **high** for IO read/write operation.

Q1.83 In which lines the 8085 processor gives the output of IO port address during IO read/write operation?

When the processor executes an IO read or write cycle, 8-bit port address is sent out both on low order address bus and high order address bus. This facility offers a flexibility for system designer to use either low-order address lines or high-order address lines for addressing ports and generating chip select signals for IO devices.

Q1.84 When the 8085 processor checks for an interrupt?

In the second T-state of the last machine cycle of every instruction, the 8085 processor checks whether an interrupt request is made or not.

Q1.85 What is interrupt acknowledge cycle?

The interrupt acknowledge cycle is a machine cycle executed by 8085 processor after acceptance of the interrupt to get the address of the interrupt service routine in-order to service the interrupting device.

Q1.86 What will be the status of the processor during bus idle cycle?

During bus idle cycle, the status signals S_0 and S_1 are both asserted **low** and data, address and control pins are driven to **high impedance** state. Also, the processor will not sample the READY signal.

Q1.87 How the slow peripherals are interfaced with 8085 processor?

The slow peripherals require longer read/write time than allowed by the processor. Hence to interface slow peripherals, an extra hardware should be designed so that it introduces required number of wait states in machine cycles between T_1 and T_2. An alternate solution is to interface the slow peripherals using ports.

Q1.88 When is the READY signal sampled by the processor?

The 8085 processor samples or checks the READY signal at the second T-state of every machine cycle.

Q1.89 What are wait states?

The T states introduced between T_2 and T_3 of a machine cycle by the slow peripherals (to get extra time for read/write operation) are called wait states.

Q1.90 When the 8085 processor will enter wait state?

The 8085 processor will check the READY signal at the second T-state of a machine cycle. If the READY is tied **low** at this time, then it will enter into wait state (i.e., after second T-state). The processor will come out of wait state only when READY is again made **high**.

Q1.91 What is the difference between wait state and bus idle condition?

During bus idle condition, the tristate pins of the processor are driven to **high impedance** state, but during wait state they are in normal states (either **low** or **high**). The READY is not sampled during bus idle condition but it is sampled during wait state.

Q1.92 How many instructions are available in 8085 instruction set?

The 8085 instruction set consists of 74 basic instructions and 246 total instructions.

Q1.93 What is an Interrupt ?

Interrupt is a signal sent by an external device to the processor so as to request the processor to perform a particular task or work.

Q1.94 How are interrupts classified ?

There are three methods of classifying interrupts.

Method I : The interrupts are classified into Hardware and Software interrupts.
Method II : The interrupts are classified into Vectored and Non-vectored interrupt.
Method III : The interrupts are classified into Maskable and Non-maskable interrupts.

Q1.95 How does a microprocessor service an interrupt request ?

When the processor recognizes an interrupt, it saves the processor status in stack. Then it calls and executes an **I**nterrupt **S**ervice **R**outine (ISR). At the end of ISR, it restores the processor status and the program control is transferred to the main program.

Q1.96 What is the function of interrupt service routine?

For each interrupt the processor has to perform a specific job. An interrupt service routine has been developed in order to perform the operations required for a device that is interrupting the processor.

Q1.97 How are interrupts affected by system reset?

Whenever the processor or system is reset, all the interrupts except TRAP are disabled. In order to enable the interrupts, EI instruction has to be executed after a reset.

Q1.98 What are Software interrupts?

Software interrupts are program instructions. These instructions are inserted at desired locations in a program. While running a program, if a software interrupt instruction is encountered then the processor executes an interrupt service routine.

Q1.99 What is Hardware interrupt?

If an interrupt is initiated in a processor by applying an appropriate signal to an interrupt pin, then the interrupt is called Hardware interrupt.

Q1.100 What is the difference between Software and hardware interrupts?

Software interrupt is initiated by the main program, but a hardware interrupt is initiated by an external device.

In 8085, the software interrupt cannot be disabled or masked but the hardware interrupt except TRAP can be disabled or masked.

Q1.101 What are vectored and non-vectored interrupt?

When an interrupt is accepted, if the processor control branches to a specific address defined by the manufacturer, then the interrupt is called vectored interrupt.

In non-vectored interrupt, there is no specific address for storing the interrupt service routine. Hence, the interrupting device should give the address of the interrupt service routine.

Q1.102 List the software and hardware interrupts of 8085.

Software interrupts : RST 0, RST1, RST 2, RST 3, RST 4, RST 5, RST 6 and RST 7.
Hardware interrupts : TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR.

Q1.103 What is TRAP?

TRAP is a non-maskable interrupt of 8085. It is not disabled by processor reset or after recognition of interrupt.

Q1.104 Does HOLD has higher priority than TRAP or not?

The interrupts including TRAP are recognized only if the HOLD is not valid, hence TRAP has lower priority than HOLD.

Q1.105 What is masking and why it is required?

Masking is preventing the interrupt from disturbing the current program execution. When the processor is performing an important job (process) and if the process should not be interrupted then all the interrupts should be masked or disabled.

In processor with multiple interrupts, the lower priority interrupt can be masked so as to prevent it from interrupting, the execution of interrupt service routine of higher priority interrupt.

Q1.106 When does the 8085 processor accept a hardware interrupt?

The processor keeps on checking the interrupt pins at the second T-state of the last machine cycle of every instruction. If the processor finds a valid interrupt signal and if the interrupt is unmasked and enabled then the processor accepts the interrupt. The acceptance of the interrupt is acknowledged by sending an $\overline{\text{INTA}}$ signal to the interrupting device.

Q1.107 List the type of signals that have to be applied to initiate a hardware interrupt in 8085.

The TRAP is level and edge-sensitive and so the interrupt signal has to take a **low** to **high** transition and then remain **high** until it is recognized. The RST 7.5 is edge-sensitive and so the interrupt signal has to take a **low** to **high** transition and need not remain **high** until it is recognized. The RST 6.5, RST 5.5 and INTR are level-senstive and so the interrupt signal should be **high** until the interrupt is recognized.

Q1.108 What are maskable and non-maskable interrupts of 8085?

The TRAP is non-maskable interrupt. The RST 7.5, RST 6.5 and RST 5.5 are maskable interrupts. The INTR of 8085 can also be disabled by DI instruction.

Q1.109 When will the 8085 processor disable the interrupt system ?

The interrupts of 8085 except TRAP are disabled after any one of the following operations.

- **Executing EI instruction.**
- **System or processor reset.**
- **After recognition (acceptance) of an interrupt.**

Q1.110 What is the function performed by DI instruction?

The function of DI instruction is to disable the entire interrupt system.

Q1.111 What is the function performed by EI instruction?

The EI instruction can be used to enable all the interrupts after disabling.

Q1.112 How can the interrupt INTR of 8085 be expanded?

The interrupt INTR of 8085 can be expanded upto eight interrupts using 8-to-3 priority encoder. It can also be expanded to eight interrupts using one number of 8259 (Programmable interrupt controller) or upto 64 interrupts using 8259's in cascaded mode.

Q1.113 How can we check whether an 8085 interrupt is masked or not?

The masking status of an 8085 interrupt can be obtained by executing RIM instruction. When RIM instruction is executed, a 8-bit data is loaded in the accumulator. The bits B_0, B_1 and B_2 will give the masking status of RST 5.5, RST 6.5 and RST 7.5 respectively. If this bit is 1, then the corresponding interrupt is masked, otherwise it is unmasked.

Q1.114 How can we check the interrupt request pending status of 8085 interrupt?

The pending status of an 8085 interrupt can be obtained by executing RIM instruction. When the RIM instruction is executed an 8-bit data is loaded in accumulator. The bits B_4, B_5 and B_6 will give the pending status of RST 5.5, RST 6.5 and RST 7.5 respectively. If this bit is 1, then the interrupt is pending, otherwise it is not pending.

Q1.115 What is vectoring?

Vectoring is the process of generating the address of interrupt service routine to be loaded in program counter.

Q1.116 How are vector addresses generated for hardware interrupts of 8085?

For the hardware interrupts TRAP, RST 7.5, RST 6.5 and RST 5.5 the vector addresses are generated by the processor itself. These addresses are fixed by the manufacturer.

Q1.117 How are vector addresses generated for software interrupts of 8085?

For the software interrupts RST 0 to RST 7, the vector addresses are generated internal to the processor. These vector addresses are fixed by the manufacturer.

Q1.118 How can the hardware interrupt of 8085 be masked or unmasked?

The masking or unmasking of RST 7.5, RST 6.5 and RST 5.5 interrupts can be performed by moving an 8-bit data to accumulator and then executing SIM instruction. The format of the 8-bit data is shown in Fig. Q1.118.

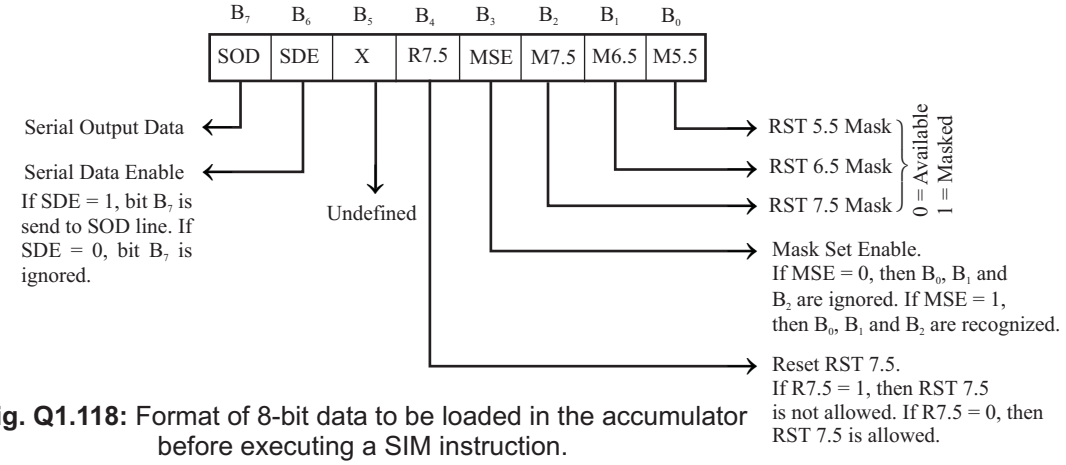


Fig. Q1.118: Format of 8-bit data to be loaded in the accumulator before executing a SIM instruction.

Q1.119 What is synchronous data transfer scheme?

In synchronous data transfer scheme, the processor does not check the readiness of the device after a command has been issued for read/write operation. In this scheme the processor will request the device to get ready and then read/write to the device immediately after the request. In some synchronous schemes a small delay is allowed after the request.

Q1.120 How can the status of maskable interrupts be read in 8085 processor?

The status of hardware interrupts like interrupt request pending or not, interrupts enabled or not, and masked or unmasked can read from accumulator after executing RIM instruction. When RIM instruction is executed an 8-bit data is loaded in accumulator which can be interpreted as shown in Fig. Q1.120.

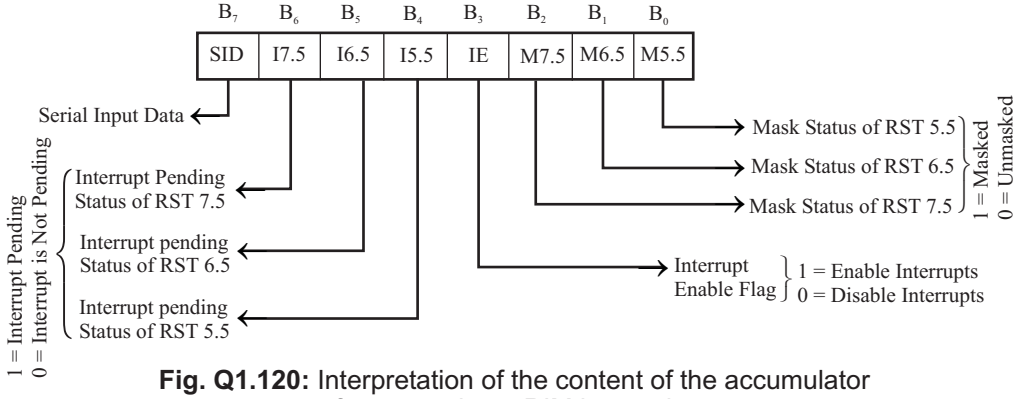


Fig. Q1.120: Interpretation of the content of the accumulator after executing a RIM instruction.

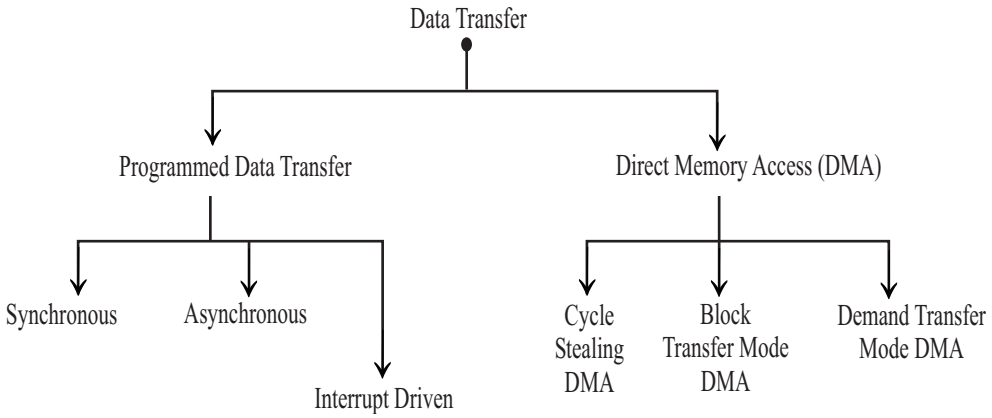
Q1.121 How is a vector address generated for the INTR interrupt of 8085?

For the INTR interrupt, the interrupting device has to place either RST opcode or CALL opcode followed by a 16-bit address. If RST opcode is placed, then the corresponding vector address is generated by the processor. In case of CALL opcode the given 16-bit address will be the vector address.

Q1.122 What is data transfer scheme and what are its types?

The data transfer scheme refers to the method of data transfer between the processor and peripheral devices.

The different types of data transfer schemes are shown below :



Q1.123 What is asynchronous data transfer scheme?

In asynchronous data transfer scheme, first the processor sends a request to the device for read/write operation. Then the processor keeps on polling the status of the device. Once the device is ready, the processor executes a data transfer instruction to complete the process.

Q1.124 How is DMA initiated?

When the IO device needs a DMA transfer, it will send a DMA request signal to the DMA controller. The DMA controller inturn sends a HOLD request to the processor. When the processor receives a HOLD request, it will drive its tristate pins to **high impedance** state at the end of current instruction execution and sends an acknowledge signal to the DMA controller. Now the DMA controller will perform DMA transfer.

Q1.125 What are the different types of DMA?

The different types of DMA data transfer are cycle stealing (or single transfer) DMA, Block transfer (or Burst mode) DMA and Demand transfer DMA.

Q1.126 What is cycle stealing DMA?

In cycle stealing DMA (or single transfer mode) the DMA controller will perform one DMA transfer in between the instruction cycles (i.e., In this mode the execution of one processor instruction and one DMA data transfer will take place alternatively).

Q1.127 What are block and demand transfer modes DMA?

In block transfer mode, the DMA controller will transfer a block of data and relieve the bus to the processor. After sometime another block of data is transferred by the DMA and so on.

In demand transfer mode the DMA controller will complete the entire data transfer at a stretch and then relieve the bus to the processor. ***(AU, Nov/Dec' 19, 2 Marks)***

Q1.128 List two major difference between INTR and the other hardware interrupts.

1. INTR is non-vectored interrupt but all other hardware interrupts are vectored interrupt.
2. INTR is expandable using 8-to-3 priority encoder or using INTEL 8259 programmable interrupt controller, but other hardware interrupts can not be expanded.

1.8 EXERCISES

I. Fill in the blanks with appropriate words

1. A digit of the binary number or code is callled _____ .
2. The group of conducting lines that carry control signals is called _____ bus.
3. The _____ state is used to keep the device electrically isolated from the system.
4. The third generation microprocessors were designed using _____ technology.
5. Transfering different information at different well defined times through the same lines is called _____.
6. 1 mil is equivalent to _____ inch.
7. The _____ and _____ signals of 8085 are used for serial data communication between 8085 and any serial device.
8. The _____ register of 8085 points to the next instruction to be executed.
9. _____ flag is set to 1 if the most significant bit of the result is one.
10. A semiconductor memory with n address lines and m data lines has a memory capacity of _____ bits.

11. The memory which retains data even after the power supply to the chip is switched OFF is called _____ memory.
12. RAM is a _____ memory.
13. The ROM which can be erased by passing electrical current or UV light is called _____.
14. The memory which requires frequent refreshing is called _____.
15. The 1024 × 4 bit memory chip has _____ address lines and _____ data lines.
16. The $2^{11} \times 8$ bit memory chip has _____ number of memory locations and each location can store _____ number of bits.
17. The number of address lines needed to address a location in 8 kB memory is _____.
18. The _____ decoder is used to generate four chip select signals.
19. If the starting address of 4 kB RAM chip is mapped to 2000_H the end address will be _____.
20. The address range of 32 kB EPROM chip interfaced to 8085 system is from _____ to _____ if $\overline{CS} = \overline{A}_{15}$.
21. The number of 256 × 4 bits RAM chips required to make up 1 kB memory is _____.
22. The _____ is the data transfer scheme which enables direct data transfer between memory and IO devices.
23. Only _____ and _____ instructions can be used for data transfer between the IO devices and the processor in IO mapped devices.
24. The process of interrupting the normal program execution to carry out a specific task is referred to as _____.
25. The interrupts initiated by external hardware by sending an appropriate signal to the interrupt pin of the processor are called _____ interrupts.
26. The interrupt for which interrupt service subroutine address is predefined is called _____ interrupt.
27. The interrupt which cannot be rejected by the processor is called _____ interrupt.
28. The vector address of RST 5.5 is _____.
29. The non-maskable interrupt of 8085 is _____.
30. The second highest priority interrupt of 8085 is _____.
31. The _____ interrupt of 8085 is used during power failure.

Answers

1. bit	9. Sign	17. 13	25. hardware
2. control	10. $2^n \times m$	18. 2 to 4	26. vectored
3. high impedance	11. non-volatile	19. $2FFF_H$	27. non-maskable
4. High Density MOS(HMOS)	12. volatile	20. 0000_H, $7FFF_H$	28. $002C_H$
5. multiplexing	13. EPROM	21. 8	29. TRAP
6. 10^{-3}	14. dynamic RAM	22. direct memory access (DMA)	30. RST 7.5
7. SID, SOD	15. 10, 4	23. IN, OUT	31. TRAP
8. Program Counter(PC)	16. 2048, 8	24. interrupt	

II. State whether the following statements are True/False.

1. The address bus is bidirectional.
2. The CPU bus is directly connected to the microprocessor.
3. The high impedance state is an electrical open-circuit condition .
4. The NMOS technology offers faster speed and higher density than HMOS technology.
5. Registers can be read/written faster than memory chips.
6. The 8085 has an internal clock oscillator.
7. The 8085 interrupts RST 7.5, RST 6.5 and RST 5.5 are non-maskable interrupts.
8. To reset the 8085 microprocessor the $\overline{\text{RESET IN}}$ pin should be held low for atleast three clock pulses.
9. The stack pointer (SP) holds the address of the stack top.
10. The registers of microprocessors can be accessed faster than the main memory.
11. The primary memory is slower than secondary memory.
12. The volatile memory loses its data when the power supply to the chip is switched OFF.
13. The EPROM is a non-volatile memory.
14. All the semiconductor memories are non-destructive readout memory.
15. PROMs are many time programmable by the user.
16. Static RAM can pack more bits than dynamic RAM in a given physical area.
17. The dynamic RAM (DRAM) is faster than static RAM and consumes less power.
18. The flash memory is a type of non-volatile RAM.
19. The unused address lines of microprocessor are generally used for generating chip select signals.
20. All the memory/peripheral ICs always remain in active state.
21. IO devices can be directly connected to microprocessors.
22. In memory mapped I/0 technique, both memory and IO devices are treated in a same way.
23. In I/O mapped ports, the data can be moved from any registers to the ports and vice versa.
24. A separate decoder is required to generate the chip select signals for IO mapped devices.
25. In polling technique, the processor has to check the readiness of the device periodically for data transfer.
26. In interrupt driven data transfer scheme, the external device interrupts the processor only when the data is ready.
27. The Interrupt Service Subroutine (ISR) address is predefined by the processor manufacture for non vectored interrupts.
28. The processor should compulsorily accept all interrupts any time.
29. The software interrupts cannot be masked.
30. The vector address of 8085 software interrupt RSTn is equivalent to n×8.
31. All the hardware interrupts are disabled when the 8085 processor is reset.
32. INTR interrupt of 8085 is the highest priority interrupt.

Answers

1. False	7. False	13. True	19. True	25. True	31. False
2. True	8. True	14. True	20. False	26. True	32. False
3. True	9. True	15. False	21. False	27. False	
4. False	10. True	16. False	22. True	28. False	
5. True	11. False	17. True	23. False	29. True	
6. True	12. True	18. True	24. True	30. True	

III. Choose the right answer for the following questions.

1. ***Microprocessors are intended to be a _____ computer.***

   a) general-purpose ***b)*** special purpose ***c)*** hybrid ***d)*** analog

2. ***Group of 4-bits is called***

   a) byte ***b)*** nibble ***c)*** word ***d)*** double word

3. ***What would be the total memory capacity of a microprocessor with 10 address lines***

   a) 1 MB ***b)*** 1 GB ***c)*** 1 KB ***d)*** 512 MB

4. ***The first 8-bit processor introduced by INTEL is***

   a) 8080 ***b)*** 8008 ***c)*** 8085 ***d)*** 8086

5. ***Which of the following is used to store temporary programs and data?***

   a) EPROM ***b)*** ROM **c)** RAM ***d)*** all the three

6. ***The 1-bit register provided to store the results of certain program instructions is***

   a) status register ***b)*** instruction register ***c)*** program counter ***d)*** flag

7. ***Which of the following is not a 16-bit processor?***

   a) 8086 ***b)*** 80186 ***c)*** 8088 ***d)*** 8096

8. ***Which of the following signals is used to demultiplex the address/data lines in 8085 and 8086 processors?***

   a) DT/$\overline{R}$ ***b)*** ALE ***c)*** SOD ***d)*** READY

9. ***The maximum internal clock frequency of 8085A microprocessor is***

   a) 5.03 MHz ***b)*** 6 MHz ***c)*** 10.6 MHz ***d)*** 3.03 MHz

10. ***The total memory capacity of 8085 processor is***

    a) 64 KB ***b)*** 1 MB ***c)*** 64 MB ***d)*** 10 MB

11. ***Which of the following 8085 signals are used for DMA operation?***

    a) $\overline{RD}$, $\overline{WR}$ ***b)*** $\overline{RESET\ IN}$, $\overline{RESET\ OUT}$ ***c)*** HOLD, HLDA ***d)*** SOD, SID

12. ***When initiated, certain interrupts can be delayed or rejected but when allowed, the program execution starts from a fixed location. Such an interrupt is known as,***

    a) non maskable and non vectored ***b)*** maskable and non vectored
    c) non maskable and vectored ***d)*** maskable and vectored

13. ***The vector address for the 8085 interrupt TRAP is,***

    a) 0028_H ***b)*** 0020_H ***c)*** 0024_H ***d)*** 0000_H

14. ***Which of the following is volatile memory?***

    a) RAM ***b)*** ROM ***c)*** PROM ***d)*** EPROM

15. What is the range of address (in hexadecimal) for 16 KB memory?

a) 0000_H to $FFFF_H$ *b)* 0000_H to $03FF_H$ *c)* 0000_H to $3FFF_H$ *d)* 0000 to $7FFF_H$

16. In 8085 processor, how many 1024 × 4 bits memory chips are required for a system memory of 4 kB?

a) 4 *b)* 8 *c)* 12 *d)* 16

17. The starting address of a 32 kB memory chip in 8085 processor is 0000H. What is the end address?

a) $FFFF_H$ *b)* $7FFF_H$ *c)* $3FFF_H$ *d)* $1FFF_H$

18. A memory chip has 11 address lines and 4 data lines. How many such chips are required to design a memory of 4 kB?

a) 4 *b)* 8 *c)* 16 *d)* 32

19. In 8085 microprocessor, a 4 kB EPROM and 4 kB RAM are selected when $A_{15} = A_{14} = A_{13} = A_{12} = 0$ and $A_{15} = A_{14} = A_{13} = A_{12} = 1$ respectively. What is the unused address space?

a) 1000_H to $EFFF_H$ *b)* 2000_H to $DFFF_H$ *c)* 4000_H to $FFFF_H$ *d)* 6000_H to $CFFF_H$

20. An 8 kB EPROM and 8 kB RAM are interfaced to 8085 microprocessor. Which of the following can be possible correct address range for this memory devices,

i) EPROM = 0000_H to $1FFF_H$
RAM = 2000_H to $3FFF_H$

ii) EPROM = 0000_H to $2FFF_H$
RAM = 3000_H to $3FFF_H$

iii) EPROM = 0000_H to $0FFF_H$
RAM = 1000_H to $2FFF_H$

iv) EPROM = 0000_H to $1FFF_H$
RAM = $D000_H$ to $FFFF_H$

a) i *b)* iii *c)* i and iv *d)* ii and iv

21. A 16 kB RAM in an 8085 system has address range 4000_H to $7FFF_H$. If the unused address lines are used to generate the chip select signals the $\overline{CS}$ = ?

a) $\overline{CS} = \overline{A}_{15}\overline{A}_{14}$ *b)* $\overline{CS} = \overline{A}_{15}A_{14}$ *c)* $\overline{CS} = A_{15}$ *d)* $\overline{CS} = A_{14}$

22. The interrupts whose request can be either accepted or rejected by the processor are called as

a) vectored interrupts *b)* Non-vectored interrupts
c) maskable interrupts *d)* non-maskable interrupts

23. Which of the following is not a hardware interrupt of 8085?

a) TRAP b) RST 7.5 c) INTR *d)* RST n

24. The vector address of the 8085 software interrupt RST 3 is

a) 0008_H b) $000C_H$ c) 0010_H *d)* 0018_H

25. Which of the following signal on 8085 TRAP line initiates the interrupt?

a) b) c) *d)* none

26. Which of the following 8085 interrupt is not a level sensitive interrupt?

a) RST 7.5 b) RST 6.5 c) RST 5.5 *d)* INTR

27. The 8085 receives both DMA request and an interrupt on TRAP pin. Which of the following statement is true?

a) It recognizes TRAP first and then DMA request

b) It recognizes DMA request first and then TRAP

c) It recognizes both TRAP and DMA request simultaneously.

d) It rejects both requests and execute the next instruction in the main program.

28. In the 8085 microprocessor, the RST 5 instruction transfers the program control to the following location.

a) 0010_H b) 0018_H c) 0020_H ***d)*** 0028_H

Answers

1. a	8. b	15. c	22. c
2. b	9. d	16. b	23. d
3. c	10. a	17. b	24. d
4. b	11. c	18. a	25. c
5. c	12. d	19. a	26. a
6. d	13. c	20. c	27. b
7. d	14. a	21. c	28. d

IV. Answer the following questions.

E1.1 What is meant by addressability of a microprocessor?

E1.2 State the significance of clock pulse in microprocessor based system.

E1.3 List some of the 4-bit microprocessors with specifications.

E1.4 Define the term speed power product(SPP).

E1.5 Mention the packing density of NMOS and HMOS technology

E1.6 Define the term MIPS.

E1.7 Draw the basic functional blocks of a microprocessor.

E1.8 What is meant by active low signal? Explain with an example.

E1.9 Write down the significance of bus status signals $IO/\overline{M}$, S_0 and S_1.

E1.10 What is the use of signals SOD and SID in 8085?

E1.11 Define maskable and non-maskable interrupts.

E1.12 What is meant by hardware interrupt?

E1.13 What is meant by software interrupt?

E1.14 What is meant by vectored and non-vectored interrupts?

E1.15 Write down the vector address of all the interrupts of 8085.

E1.16 When power on, how does the CPU know the starting address of the first instruction it has to execute? What is that first instruction? why?

E1.17 What is the use of stack pointer in 8085 microprocessor?

E1.18 Why the program counter and stack pointer of 8085 are 16-bit registers?

E1.19 A semiconductor memory has 16 address lines and 8 data lines. What is the capacity of the memory?

E1.20 Show the memory interfacing of 32 kB RAM and 32 kB ROM with 8085 microprocessor.

E1.21 Write an 8085 ALP to enable all the interrupts of 8085.

E1.22 The 8085 processor is executing the following ISR of RST 6.5. Rearrange the program to allow the higher priority interrupt RST 7.5 to interrupt the processor while execution of RST 6.5 ISR.

```
;ISR of RST 6.5
     MVI  A,00H
Back: INR  A
     CPI  A,0FFH
     JNZ  BACK
     RET
```